**Warehouse Management System**

**Senior Project Presented to the Faculty of Science**

**Department of Computer Sciences**

**Al Maaref University**

**In Partial Fulfillment of the Requirements for the Degree**

**Bachelor of Science**

**Computer Sciences**

**Ali Shokor-10121433**

Fall 2024-2025

|  |
| --- |
|  |
|  |
|  |
|  |

# Dedication

Mainly, we dedicate this work to God our source of inspiration, wisdom, and knowledge who gave us all the faith and strength to complete this project.

To both of our families who stood beside us throughout all of this road and gave us the chance to become one of those who would leave an impact for the future.

And finally, to our Al-Maaref University Family, from instructors to administrators and to our friends whom without any of them we wouldn’t reach this stage of completing our degree.

# Acknowledgments

1. We extend our heartfelt gratitude to the Faculty of Science at Al-Maaref University, particularly Professor Jalal Jomaa. Throughout our academic journey, his unwavering commitment to excellence has inspired us to push the boundaries of our knowledge and capabilities.
2. Special thanks are due to Dr. Imad Moukadem, the old dean of FOS, whose insightful advice and unwavering support have been instrumental in completing our major and fostering our intellectual growth throughout all of our journey.
3. Special thanks are due to Dr. Abed Al-Safadi the head of the department who stand beside us where there is no dean for the department, whose insightful advice and unwavering support have been instrumental in completing our major and fostering our intellectual growth throughout all of our journey.
4. We are deeply indebted to Dr. Fatima Abdallah, our supervisor, for her invaluable feedback and encouragement. Her guidance and expertise have propelled us forward, empowering us to navigate challenges with resilience and determination.
5. To our families and friends, we express profound gratitude for their unwavering belief in us and their unconditional support. Their love and encouragement have been a constant source of motivation, guiding us through the highs and lows of our academic journey.
6. To all those who have supported us along the way, your contributions have been immeasurable, and we are profoundly grateful for your role in our success.

# Table of Contents

Chapter I Introduction ............................................................................................................. 10

1.1. Overview ...................................................................................................................10

1.2. Problem Statement ....................................................................................................10

1.3. Suggested solution.....................................................................................................11

1.4. Gantt chart ................................................................................................................. 11

Chapter II State of the Art ........................................................................................................ 15

2.1. Introduction ............................................................................................................... 15

2.2. Similar Applications .................................................................................................. 15

2.2.1. Toters Website system…........................................................................................ 16

2.3. Available Technologies ............................................................................................. 21

2.3.1. Backend Frameworks ........................................................................................ 21

2.3.2. Frontend Frameworks.........................................................................................24

2.3.3. Database Frameworks ........................................................................................ 25

2.4. Conclusion ................................................................................................................. 26

Chapter III Requirement Analysis ...........................................................................................28

3.1. Introduction ............................................................................................................... 28

3.2. High-level Business Objectives ................................................................................ 28

3.3. Functional Requirements........................................................................................... 29

3.4. Non-Functional Requirements .................................................................................. 30

3.4.1. Quality NFRs ..................................................................................................... 31

3.4.2. Performance ....................................................................................................... 31

3.4.3. Constraints ......................................................................................................... 31

3.5. Conclusion ................................................................................................................. 32

Chapter IV Application Modeling and Design ........................................................................ 33

4.1. Introduction ............................................................................................................... 33

4.2. UML Diagrams ......................................................................................................... 33

4.2.3. Use Case Diagram.............................................................................................. 33

4.2.3. Scenarios ............................................................................................................ 35

4.2.3. Activity Diagrams .............................................................................................. 36

4.3. Database Design ........................................................................................................39

4.3. Conclusion ................................................................................................................. 43

Chapter V Application Implementation ................................................................................ 44

5.1. Introduction ............................................................................................................... 44

5.2. Used Technologies .................................................................................................... 44

5.2.1. Used Backend Technology ................................................................................ 44

5.2.2. Used Frontend Technology ................................................................................ 44

5.3. Application Screenshots ............................................................................................ 45

5.4. Code Screenshots ...................................................................................................... 53

5.6. Conclusion .................................................................................................................59

Chapter VI General Conclusion .............................................................................................. 60

# Table of Tables

Table 1 Table of Dependencies................................................................................................

Table 2 Features of Similar Websites ......................................................................................

Table 3 Features of Database Frameworks ..............................................................................

Table 4 Features of Backend Frameworks ...............................................................................

Table 5 Features of Frontend Frameworks ..............................................................................

Table 6 Quality Non-Functional Requirements .......................................................................

Table 7 Constraints Non-Functional Requirements .................................................................

Table 8 Booking an Appointment Scenario for Registered User ............................................

Table 9 Booking a Movie Scenario for the cashier ..................................................................

# List of abbreviations

WSGI -Web Server Gateway Interface

HTML - Hypertext Markup Language

JSX - JavaScript XML

MVC – Model-View-Controller

PHP - Hypertext Preprocessor

UML - Unified Modeling Language

ER – Entity Relationship

PO- Purchase Order

CO – Customer Order

WMS-Warehouse Management System

**Abstraction**

The Warehouse Management System (WMS) is a comprehensive digital solution designed to streamline operations for modern warehouses, addressing inefficiencies in inventory management, order processing, and resource allocation. The system supports four primary user roles: the invoice controller , responsible for managing purchase orders (POs), suppliers, and product attributes; the receiver , tasked with verifying deliveries and updating inventory levels; the store admin , who oversees customer orders (COs) and ensures timely fulfillment; and the auditor , who monitors stock-level logs and ensures compliance. Key functionalities include user authentication, allowing all users to securely sign in using their username and password. The invoice controller can view item quantities, create POs, add new products, edit product attributes, view supplier-specific products, create new suppliers, and manage low-stock alerts. The receiver collaborates with the invoice controller to add requested products to POs, view PO details, enter received quantities, and upload physical invoices. The store admin manages customer orders by viewing, adding, removing, or editing items, while also having the ability to adjust inventory quantities in collaboration with the auditor, who ensures transparency by logging reasons for quantity adjustments. Additionally, the auditor has exclusive access to stock-level logs for monitoring inventory changes. By leveraging modern web technologies, the WMS eliminates manual inefficiencies, enhances accuracy, and improves customer satisfaction through real-time tracking, automated processes, and role-based access control. Future enhancements aim to integrate IoT devices for monitoring warehouse conditions, implement AI-driven analytics for demand forecasting, and connect the system to external APIs for seamless integration with third-party logistics providers and e-commerce platforms, ensuring scalability and innovation for years to come.

**Introduction**

## 1.1. Overview

Our Warehouse Management System (WMS) is designed to streamline and optimize warehouse operations by improving inventory management, order processing, and overall efficiency. The system provides real-time tracking of stock levels, automates key warehouse tasks, and minimizes errors associated with manual processes. It includes features such as barcode scanning, automated stock updates, order picking and packing management, and integration with other supply chain systems. With an intuitive user interface and powerful analytics, the WMS enhances decision-making, reduces operational costs, and improves customer satisfaction. By implementing this system, businesses can achieve greater accuracy, faster order fulfillment, and better utilization of warehouse space, ensuring a more efficient and scalable supply chain.

## 1.2. Problem Statement

Many warehouses struggle with inefficiencies in managing product imports and exports, processing purchase and customer orders, handling invoices, and monitoring stock levels. Manual processes and outdated systems often lead to inventory discrepancies, shipment delays, order fulfillment errors, and financial record mismanagement. Additionally, the lack of real-time stock monitoring increases the risk of product shortages, disrupting operations and affecting customer satisfaction. Without an automated system, warehouse staff face challenges in tracking shipments, processing invoices accurately, and ensuring timely restocking. These inefficiencies result in increased operational costs, wasted time, and reduced overall productivity. To address these challenges, a Warehouse Management System (WMS) is essential to automate and optimize warehouse operations, improving accuracy, efficiency, and overall supply chain performance.

## 1.3. Suggested solution

## The proposed Warehouse Management System (WMS) includes several key solutions to enhance efficiency, including managing the import and export of products, handling purchase and customer orders, streamlining invoice management, and implementing stock alerts. The system automates tracking of incoming and outgoing shipments, ensuring real-time inventory updates and minimizing errors. It efficiently processes purchase orders by integrating supplier data, stock levels, and order approvals, ensuring smooth restocking. Customer order management is optimized through automated order fulfillment, shipment tracking, and error reduction. Additionally, the system simplifies invoice management by generating accurate invoices, tracking payments, and maintaining financial records. To prevent stock shortages, the WMS includes an alert system that notifies warehouse managers when product quantities are low, enabling timely reordering. These features collectively improve operational efficiency, reduce manual workload, and enhance overall warehouse productivity.

## 1.4. Gantt chart

A commonly utilized project management instrument, the Gantt chart illustrates task scheduling over time through a visual format. Its horizontal axis showcases a project's timeline, typically divided into segments like days, weeks, or months. On the vertical axis, it enumerates the necessary tasks or activities for project completion. Each task is depicted by a horizontal bar indicating its start date, duration, and end date. This graphical representation enables project managers and team members to swiftly discern task dependencies, allocate resources efficiently, track project advancement, and make adjustments to the timeline.

|  |  |  |
| --- | --- | --- |
| Task Number | Task Description | Task Duration  (Days) |
| A | Requirements gathering and analysis | 16 |
| B | System design | 14 |
| C | Data Base Design | 15 |
| D | Back End Development | 16 |
| E | Front End Development | 16 |
| F | Integration and Testing | 13 |

Table 1 Table of Dependencies

Table 1 shows the tasks done for this project with the duration for each task and its dependencies, and the tasks are as follows:

* Task A: Requirement Analysis

The information collected was analyzed and put into requirements that must be done accordingly to finish the website.

* Task B: System Design

This stage involved designing the whole design of the website.

* Task C: Database Design

Database Design is an essential component of software development, guaranteeing that data is stored effectively, accurately, and can be quickly accessed.

* Task D: Backend implementation

The backend implementation phase in software development involves creating the server-side logic, database interactions, and integration with other services necessary to provide a complete application.

* Task E: Frontend Implementation

The frontend implementation phase in software development involves creating the user interface and user experience of an application.

## ➢ Task F: Testing

The testing phase in software development is a critical stage where the application is evaluated to ensure it functions correctly, meets the specified requirements, and is free of defects.

## 

Figure 2 shows the Gantt Chart summarizing the tasks in table 1 that should be done to complete the development of the project. It is added as a landscape to make it clearer.

And in the following part of this report, chapter 2 will include a comparison between our application and other similar ones in addition to the available technologies from the backend, frontend, and database frameworks. Chapter 3 will have the requirement analysis part like the high-level objective with the functional and non-functional requirements. Moving on to chapter 4, all the diagrams that describes the project well will be listed and explained. Moreover, chapter 5 has more details about the website itself with some screenshots explaining. Finally, chapter 6 will be a summary concluding all of the report, and future insights for improving our project.

Create customer order

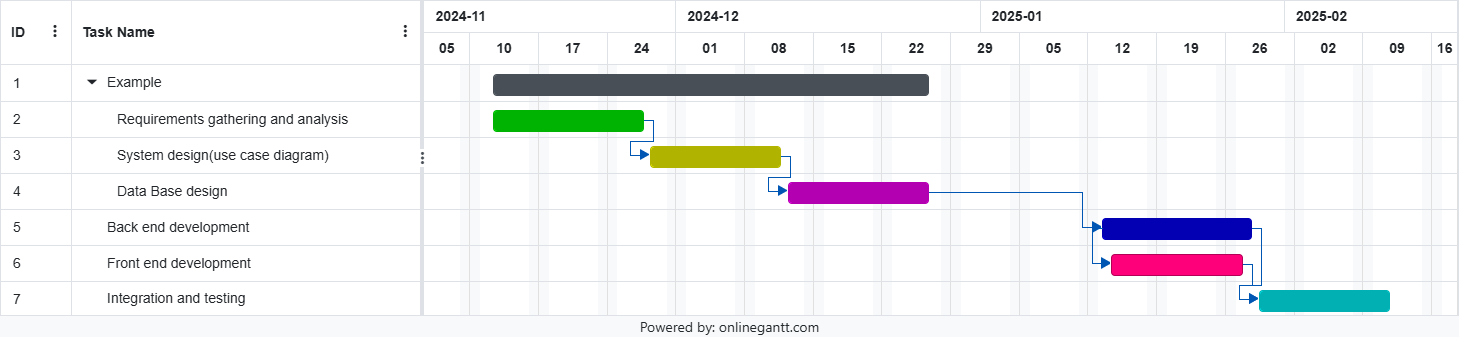


Figure 2 Gantt Chart

**Chapter II: State of the Art**

## 2.1. Introduction

In this chapter, we will explore various websites that are similar to the one we are developing. These websites will serve as a source of inspiration for creating an innovative platform that aims to enhance user experiences and introduce new features, while also providing resources and information on real estate services. By examining and analyzing these existing platforms, we can gather insights and inspiration to design a website that not only meets but exceeds user expectations by incorporating essential features. Additionally, we will review some current and trending technologies for frontend-backend implementation and database modeling. We'll discuss their advantages and disadvantages to help us determine which ones best meet our needs.

## 2.2. Similar Applications

We aimed to examine the current similar websites in order to gain helpful information to our implemented website. These websites determined for us the main features that are essential to be present in any real estate website. They also gave us inspiration on how to give the user a pleasing website, what features to include and what to avoid. And since our website is made for a specific association, there are some features that won’t be available in any website other than ours such as displaying the old movies list and allowing the customer to order showing it again.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Features | Create Purchase Orders | Altering Products | Creating Customer Order | Adding invoices | Checking low stock products, with alert | Checking supplier items | Creating new product |
| Our Project | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Toters | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

Figure 3 Features of Similar Websites

## 2.2.1.Toters Company: Toters is the most famous delivery app locally in Lebanon also it has a branches named toters fresh which are an warehouses connected through their own warehouse management system and this system has many weak points that we solved at our system . the following figures are a screenshot from their own system that well explain each error they have and compare it with our system.

Also its impossible to reach any other warehouse management systems since its usually privately secured:

## 

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4-toters cusmtomer orderes view

This screenshot shows all the orderes that a store is working on and you can reach any one of the buut its issue that it takes very long time to update the status of the order also takes a time to aprrove the order since its not connected to the database of products direclty to check it faster like our system.

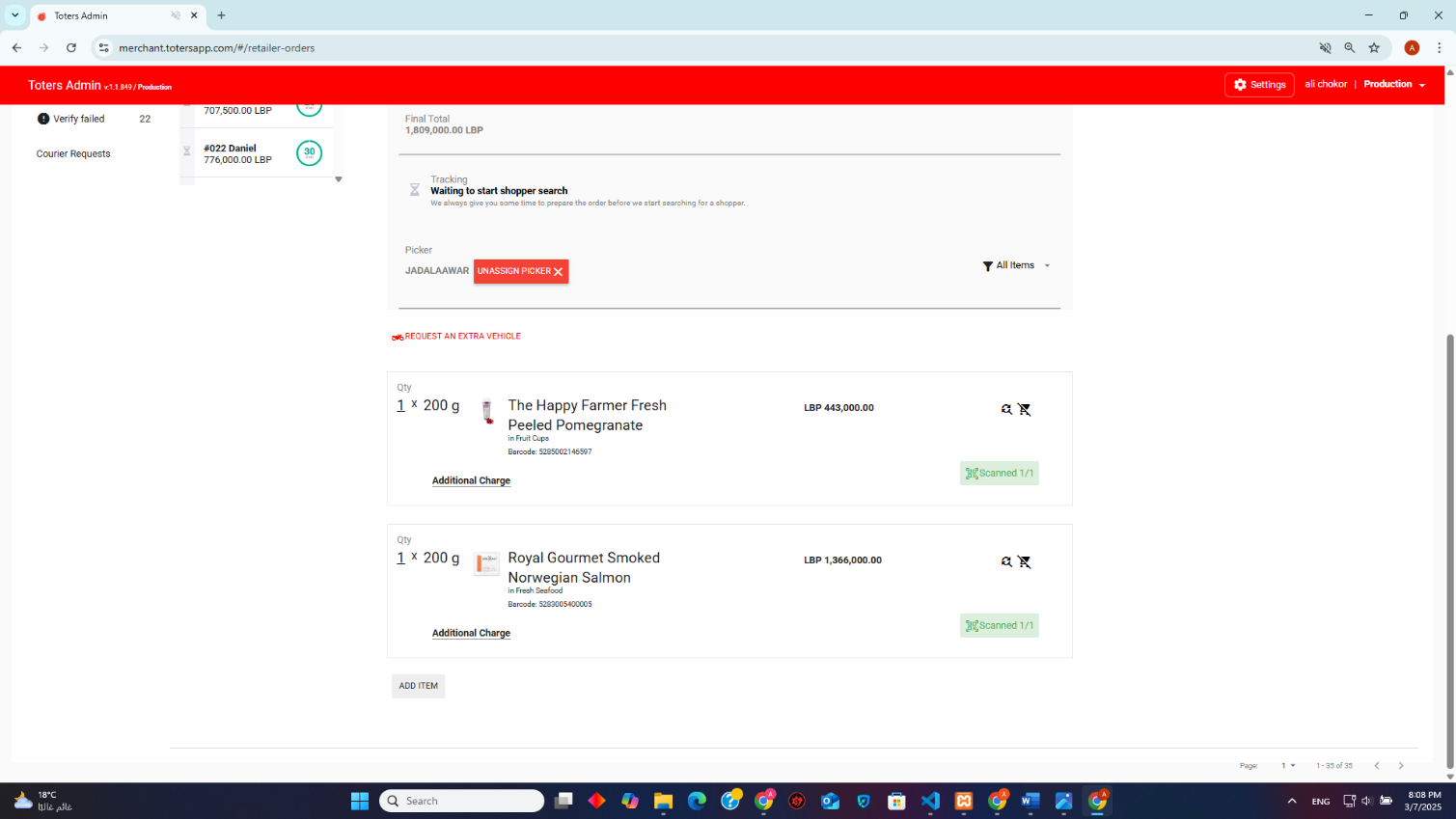


Figure5-toters customer order index 1

A screenshot of a computer

AI-generated content may be incorrect.

Figure6- toters customer order index 2

This screenshot show the customer order from inside that specifies the order information and their system is facing an issue in pricing the products and to prove that realize the price of 1 cup of pomegranate in figure 5 and when editing it to 5 pcs the price remains the same(shown in figure 6 ) until the order is received then they charge this extra on his next order while our system the price is updated at real-time.

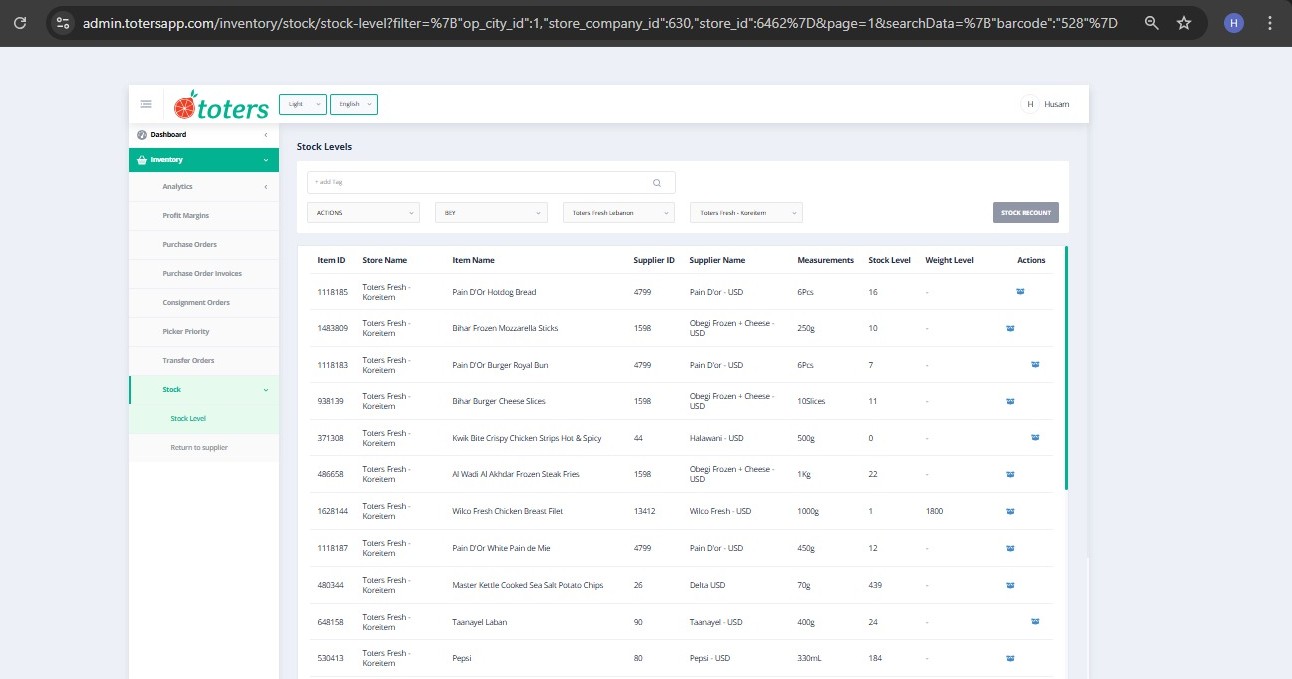
****

Figure 8-Toters stock lab

This is toters stock lab segment and its very obvious that there page is not a responsive one and there stock lab doesn’t have a stock log feature like our system

A screenshot of a computer

AI-generated content may be incorrect.

Figure 9- toters stock edit

This is the page were they edit there stock quantities were they only have the decrementing feature and cant increment any quantity if found like our system

A screenshot of a computer

AI-generated content may be incorrect.

Figure 10-Toters PO view

This is the page where all the Po are stored but the have an issue the received po’s cant be reached to investigate if there is any mistake or issue like our system is built

A screenshot of a computer

AI-generated content may be incorrect.

Figure 11- Toters PO index

This is the page where all the information of the PO are shown but is doesn’t have any receive or save buttons like our system have

## 2.3. Available Technologies

When developing a user-friendly website, it is crucial to consider the diverse range of available technologies for both backend and frontend development. Choosing the right backend and frontend technologies is essential to ensure a seamless user experience. These technologies facilitate efficient data handling, dynamic content display, and integration with external services.

The following section is meant to help us choose the best frameworks of all three database, backend, and frontend frameworks that best meets our requirements.

2.3.1 **Backend Technologies**

 The backend of a WMS is responsible for processing business logic, managing authentication, handling transactions, and integrating with external services. It must be secure, scalable, and capable of handling high-volume operations efficiently. The available technologies are :

1. **Laravel (PHP Alternative) :**

Laravel is a popular PHP framework known for its security, flexibility, and ease of development. It is often recommended for building scalable and maintainable applications, making it a solid choice for a Warehouse Management System (WMS). However, like any technology, it comes with its own set of advantages and limitations. Below is a breakdown of both:

Advantages of Using Laravel :

1. MVC Architecture:
   * Enhances  maintainability and scalability by separating logic (Model), presentation (View), and control (Controller).
2. Eloquent ORM:
   * Simplifies database interactions, making it easier to manage complex relationships in inventory and supplier data.
3. Built-in Authentication & RBAC:
   * Includes role-based access control (RBAC), ensuring users like warehouse managers, staff, and suppliers only have access to relevant features.
4. RESTful API Support:
   * Ideal for integrating with external tools like barcode scanners, third-party logistics services, and financial systems.
5. Task Scheduling & Queue Management:
   * Automates critical warehouse tasks, such as stock updates, report generation, and bulk order processing.
6. Security Features:
   * Built-in protections against SQL injection, CSRF attacks, and encrypted authentication ensure data integrity and security.
7. Flexibility and Extensibility:
   * Laravel is highly customizable, making it easy to add features like inventory tracking, order management, or notifications.

**Limitations of Laravel:**

1. Performance for Real-Time Data:
   * Not as fast as Node.js in handling real-time data processing, so it might require additional optimization for large-scale operations.
2. Hosting Environment Requirements:
   * Needs well-configured hosting environments (e.g., Nginx with PHP-FPM) to achieve optimal performance, which could add to deployment complexity.

**Spring Boot (Java Alternative):**

Spring Boot is a Java-based framework widely used for enterprise applications. It provides high performance, scalability, and strong security features, making it ideal for large-scale warehouse systems. The framework supports microservices architecture, allowing different warehouse functions to operate independently.

Despite its advantages, Spring Boot has a steep learning curve and requires significant development time. Java applications are also resource-intensive, making hosting and maintenance more expensive than PHP-based applications.

1. **Django (Python Alternative):**

Django is a Python-based framework designed for rapid development and high-security applications. It includes a built-in ORM, similar to Laravel’s Eloquent, making database management straightforward. Django’s security features are robust, providing built-in XSS and CSRF protection.

However, Django has slower performance for real-time applications compared to Node.js and requires specialized hosting environments, which can increase deployment costs.

4. **Express.js (Node.js Alternative):**

Express.js is a minimalist backend framework for Node.js, known for its speed and efficiency in handling real-time data processing. It is a strong option for inventory tracking systems that require live updates. However, Express lacks built-in ORM support, meaning database operations require third-party libraries like Mongoose. Additionally, Node.js is not well-suited for complex relational queries, making it less efficient for handling structured warehouse data.

This is a table that sums up the backend technologies comparison:

|  |  |  |  |
| --- | --- | --- | --- |
| Technology | Language | pros | cons |
| Laravel | PHP | Secure/ scalable/built-in authentication/ORM/  API-ready | Needs optimized hosting, memory consumption |
| Spring  boot | java | High performance, microservices-friendly, enterprise-level security | Complex setup, expensive hosting, steeper learning curve |
| Django | python | Security-focused, rapid development, built-in ORM | Slower for real-time applications, requires specialized hosting |

* + 1. **Frontend Technologies:**

The frontend of a WMS is responsible for displaying inventory, stock movements, and order processing details in a clear and user-friendly manner. These are some frontend technologies :

1. **HTML, CSS, and Bootstrap:**

A combination of HTML, CSS, and Bootstrap provides a lightweight, fast, and responsive UI. HTML structures the content, CSS ensures styling, and Bootstrap enhances responsiveness, making the system easily accessible across desktops, tablets, and mobile devices. Bootstrap includes pre-built UI components such as tables, navigation menus, and modals, reducing development time.

However, Bootstrap lacks advanced interactivity, meaning additional JavaScript or jQuery is needed for dynamic features. Unlike frameworks like Vue.js or React.js, Bootstrap does not support state management, so real-time inventory updates must be handled manually.

1. **Vue.js:**

Vue.js is a progressive JavaScript framework that offers two-way data binding and a component-based structure. It is ideal for real-time inventory tracking dashboards. However, Vue.js requires more setup and JavaScript expertise, increasing development complexity.

1. **React.js:**

React is known for its performance optimization and reusable components, making it suitable for large-scale applications. However, React has a steeper learning curve and requires additional libraries for state management, which increases development time.

This is a table that sums up the backend technologies comparison:

|  |  |  |
| --- | --- | --- |
| Technology | pros | Cons |
| HTML, CSS, Bootstrap | Lightweight, responsive, cross-browser compatible, fast | Lacks interactivity, no state management |
| Vue.js | Two-way data binding, reactive UI | Requires JavaScript expertise, setup complexity |
| React.js | Component-based, strong ecosystem | Steep learning curve, additional state management required |

* + 1. **Database Technologies**

The database is responsible for storing inventory records, supplier details, and warehouse transactions.

1. MySQL:

MySQL is a relational database optimized for structured warehouse data. It ensures fast query execution through indexing and partitioning and supports ACID compliance for transactional integrity. However, MySQL lacks flexibility for unstructured data, requiring schema modifications when making changes

1. PostgreSQL:

PostgreSQL supports advanced indexing and JSON storage, making it a good choice for applications requiring complex queries. However, PostgreSQL is slightly slower in read operations compared to MySQL.

1. MongoDB

MongoDB is a NoSQL database that provides schema flexibility and high-speed reads. However, it is not optimized for relational data, making it less suitable for warehouse inventory tracking.

Databases comparison table:

|  |  |  |
| --- | --- | --- |
| Technology | pros | Cons |
| MySQL | Reliable, ACID-compliant, optimized for structured data | Less flexible for schema changes |
| PostgreSQL | Advanced queries, JSON support | Slightly slower read performance |
| MongoDB | Schema flexibility, high-speed reads | Not optimized for relational data |

**Conclusion**

After evaluating various technologies, PHP Laravel emerges as the optimal backend choice for a Warehouse Management System (WMS) due to its security, scalability, and ease of development. Its MVC architecture, Eloquent ORM, and built-in authentication make it ideal for managing complex warehouse operations. For the frontend, HTML, CSS, and Bootstrap provide a lightweight, responsive, and user-friendly interface, ensuring accessibility across devices. While Bootstrap lacks advanced interactivity, it can be enhanced with JavaScript for dynamic features. For the database, MySQL is the best fit, offering reliability, ACID compliance, and fast query execution for structured warehouse data.

This combination of Laravel, HTML/CSS/Bootstrap, and MySQL ensures a secure, scalable, and efficient WMS that is easy to develop and maintain. While other technologies like Spring Boot, React.js, or MongoDB have their strengths, they often come with trade-offs in complexity or cost. The chosen stack provides the right balance of functionality, performance, and cost-effectiveness, making it the ideal solution for businesses aiming to build a robust and future-proof WMS.

# Chapter III: Requirement Analysis

## 3.1. Introduction

In the realm of software development, a "requirement" refers to a specific feature or command that a new software system should possess. It provides a detailed description of the software's operations and any restrictions it should comply with. The objective of defining a requirement is to comprehend the system's operations, recognize its users, understand their necessities, and ascertain what the system needs to do to fulfill those necessities. Requirements are categorized into functional and non-functional requirements. Functional requirements outline the software's operations and are the basis for the design of the software. Conversely, non-functional requirements specify any limitations the software might have, such as performance, security, technical constraints, project constraints, and organizational constraints. The procedure for analysing and categorizing these requirements falls under requirements analysis or requirements engineering. The main aim of requirements analysis is to produce a document, referred to as the requirements specification document, that provides a comprehensive description of the software system to be developed .

## 3.2. High-level Business Objectives

A high-level business objective is a broad goal or target that an organization aims to achieve to advance its overall mission and strategic direction. High-level business objectives are often aligned with the organization's vision and values and are designed to drive growth, profitability, innovation, sustainability, or other key outcomes.

* Allow Administrator to edit the customer order, show the purchase order, view quantities in the store and show them.
* Allow the Invoice controller to view quantities in the stock, create purchase order, add items to the stock, view whole product information, create supplier, alerts him for low stock quantities.
* Allow the auditor to edit the quantities of items in the stock, hide items in the stock.
* Allows the receiver add quantities to the purchase order, enter the number of items received, view the purchase order.

## 3.3. Functional Requirements

Functional requirements specify the capabilities and behaviors of a system, detailing what the software must accomplish under typical conditions to satisfy user needs. For developers, these requirements outline the necessary features for the system to operate correctly. They encompass both crucial features, such as authorization and payment processes, and optional enhancements that improve user experience but are not critical for the system's functionality, like the ability to leave reviews or use various filtering options .

The functional requirements we aim to fulfill are listed below.

Functional Requirements:

1. All users shall be able to sign in using their username and password.
2. The invoice controller shall be able to view the quantities of the items in the store
3. The invoice controller shall be able to create a PO by selecting the supplier
4. the invoice controller shall able to add a new item by entering the name, description/quantity, price, cost and selecting the supplier)
5. the invoice controller shall be able to edit all the product attributes
6. the invoice controller shall be able to view all the products of a specific supplier.
7. the invoice controller shall be able to create a new supplier by entering his (name/email/phone number)
8. the invoice controller shall be able to view an alert icon for low-stock products.
9. the invoice controller shall be  able to view a table containing  low-stock products
10. the invoice controller and the receiver  shall be able to add the requested product by selecting it and entering the ordered quantity
11. The receiver and invoice controller  shall be able to view the PO ( PO ID/supplier name /total cost)
12. The receiver shall enter the number of items he received and save the changes
13. The receiver shall be able to upload a copy of the received physical invoice
14. The invoice controller shall be able to confirm the PO
15. The store admin shall be able to view all the Customer orders (CO) that show the(order ID/customer name/ order status / total price)
16. The store admin shall be able to add items to the CO
17. The store admin shall be able to remove items from the CO
18. The store admin shall be able to edit quantities of items in CO
19. The store admin shall be able to view the quantities of items in the store
20. The store admin and the  auditor  shall be able to increment the quantity of an item
21. The store admin and the  auditor  shall be able to decrease the quantity of an item and show the reason
22. the auditor shall be able to access the stock-level logs

## 3.4. Non-Functional Requirements

Nonfunctional Requirements (NFRs) define the system qualities or various attributes of a system that are not directly tied to its functionality. These attributes describe how well the system performs rather than what it does. Despite their subtle nature, NFRs are crucial for the success of a system. Failure to meet NFRs can lead to systems that fall short of business, customer, market, or regulatory requirements. Properly identifying and implementing NFRs is essential. Over-specifying them can make the solution too expensive or impractical. There are three types that classify the nonfunctional requirements which are quality, performance, and constraints.

### 3.4.1. Quality NFRs

Quality NFRs, also known as quality attributes, are the characteristics that determine how a system operates rather than what it does. They encompass aspects of security, usability, interoperability, reliability, scalability, availability, portability, and scalability. Quality NFRs play a significant role in enhancing software quality, ensuring project success, and increasing user satisfaction..

Table 6 shows the quality non-functional requirements for our project.

|  |  |
| --- | --- |
| Non-Functional Requirements | Quality |
| 1. The products quantities should be shown only by the administrator and the invoice controller. | Security |
| 2. Every user should enter to his specific pages. | Security |
| 3. The password should include at least 1 capital letter, 1 small letter, 1 number, 1 special character, and its length must be greater or equal to 8 | Security |
| 4. The system should have the main language English | Usability |
| 5. The system should be able to work without crashing | Reliability |
| 6. The system should be available 24/7 | Availability |

Table 6 Quality Non-Functional Requirements

### 3.4.2. Performance

An indicator of the effectiveness with which a digital solution utilizes necessary resources to execute its functions. Essentially, it monitors the speed at which the system reacts to user requests .

### 3.4.3. Constraints

Non-Functional Requirements often include constraints, which are essentially the limitations or restrictions that define the operational boundaries of a system. These constraints, which dictate the conditions under which the system operates, can significantly influence the user experience and the overall effectiveness of the system. They add value to the system beyond its basic functionality and are critical as they directly impact the user's experience and satisfaction with the system .

Table 7 has the constraints NFRs that our system require.

|  |  |  |
| --- | --- | --- |
|  | Non-Functional Requirements | Type |
| 1. | The frontend should be implemented using HTML-5 language | Constraint |
| 2. | The interface should be designed using Tailwind CSS framework | Constraint |
| 3. | The backend should be implemented using Laravel (PHP) | Constraint |
| 4. | The system should use SQL Server as its database | Constraint |

Table 7 Constraints Non-Functional Requirements

## 3.5. Conclusion

In conclusion, understanding and meticulously defining both functional and non-functional requirements are essential for the success of any software project. Functional requirements lay the foundation by specifying what the system must do to meet user needs, while non-functional requirements ensure that the system operates efficiently, securely, and reliably under various conditions. Together, they provide a comprehensive blueprint that guides development, guarantees quality, and ensures the final product aligns with objectives and user expectations.

# Chapter IV: Application Modeling and Design

## 4.1. Introduction

The main aim of this chapter is to present and elaborate on our complex work plan comprehensively. This blueprint offers an in-depth description of all essential phases, the time designated for each activity, the exact tasks that are to be completed, and the methodical implementations depicted in the Gantt chart.

## 4.2. UML Diagrams

### 4.2.3. Use Case Diagram

In UML, use-case diagrams are used to model the behavior of a system and capture its requirements. They outline the high-level functions and scope of a system, identifying interactions between the system and its actors. These diagrams detail what the system does and how the actors interact with it, without specifying the internal operations of the system. Typically, use-case diagrams are developed in the early stages of a project and serve as a reference throughout the development process .In a use-case diagram, the system's users are represented as actors. Each actor plays a specific role in the system, referred to as a use case. Multiple actors can perform a single use-case. An actor can be an individual, such as a customer, or a machine, such as a database system or server .

The use case of our project has 4 actors, the Admin, the receiver user, the invoice controller, the auditor. All of the actors are viewers too. The use case includes nineteen use cases, and two types of relationships which are the generalization between actors to avoid repetition, association relationship between actors and the use cases, includes relationship, and finally the extends relationship.

In the following page the use case diagram of our project will be presented (figure 8), and since it has many use cases and actors, we chose to add it as a landscape to make it clearer.

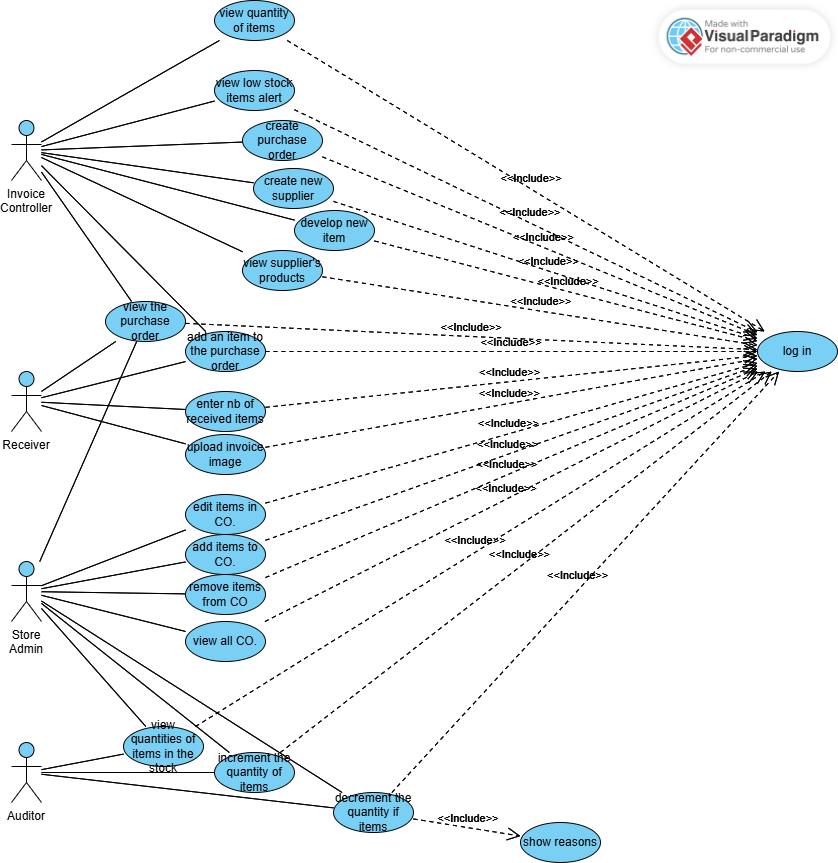


Figure12 Use Case Diagram

### 4.2.3. Scenarios

A scenario in the realm of software engineering is a detailed narrative that outlines the interaction between a user and a system. It's a sequence of specific actions that demonstrate how a user might interact with a system to accomplish a certain objective. Scenarios are typically employed during the requirements analysis stage to help illustrate the expected behavior of a proposed system. They offer a tangible example of the system's functionality under specific circumstances. Essentially, scenarios are practical examples of system usage, detailing the sequence of steps, events, and actions that transpire during the interaction . The 2 tables below show 2 different scenarios for the registered user and the cashier. Table 8 shows the scenario of creating a purchase order by the invoice controller. And table 9 shows the scenario for the creation of the customer order by the administrator.

|  |  |
| --- | --- |
| Scenario Name: | Create a purchase order |
| Participating Actors: | Invoice Controller |
| Description: | It shows how the Invoice Controller creates a purchase order. |
| Entry Conditions: | Enter to the system, and sign in as invoice controller. |
| Flow of Events: | 1. Enter the website 2. Go to the purchase orders page 3. Click on Create new Purchase Order Button. 4. Enter the supplier name. 5. Click on create purchase order. 6. Click on view button. 7. In the page of the purchase order we enter all the data we need. |
| Exit Conditions: | Click on Confirm Purchase Order button. |

Table 8. creating a purchase order.

|  |  |
| --- | --- |
| Scenario Name: | Create Customer Order |
| Participating Actors: | Administrator |
| Description: | It shows how the customer order can be created. |
| Entry Conditions: | Enter to the website and sign in as admin. |
| Flow of Events: | 1. Enter the website 2. Enter to the dash boards. 3. Click on create new customer order. 4. Enter the data. 5. And click on create order button. |
| Exit Conditions: | It takes the user to the Active Customer Orders page. |

Table 9 Creating a customer order.

### 4.2.3. Activity Diagrams

An activity diagram is a type of UML flowchart that illustrates the progression from one activity to another within a system or process. It depicts the dynamic aspects of a system and is considered a 'behavior diagram' because it outlines what should occur in the modeled system. Activity diagrams can visualize even very complex systems, making them useful for business process modeling and detailing the steps of a use case diagram within organizations. They show individual steps and their sequence, as well as the flow of data between activities. An activity diagram maps the process from the initial state to the final state, including actions, decision nodes, control flows, a start node, and an end node .

* Initial State: Portrays the beginning of a set of actions or activities and is symbolized by a full-black circle.
* Final State: Stops all control flows and object flows in an activity (or action) and is represented by an outlined black circle.

The following figure (figure 13) shows the flow of events of creating a purchase order. To do so, the Invoice Controller shall sign in and enter the website first, then go to the purchased orders page, after that click on Create Purchase Order button, then enter the supplier name, next click on create purchase order button, then click on the view button which shows the created purchase order, after that enter the data wanted, finally click on confirm purchase order.

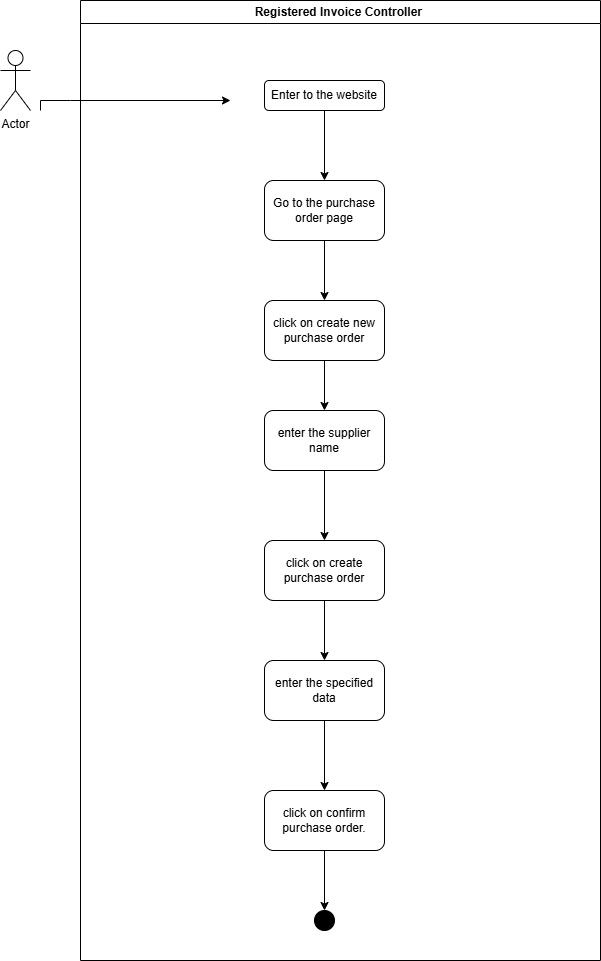


Figure 13 Activity Diagram of Invoice controller to create purchase order.

The next figure (figure 14) has the activity of Creating a customer order, first of all the administrator who is responsible for creating the customer orders must sign in, then in the dashboards he clicks on create new customer order, after that he enters the data, and finally clicks on create order button.

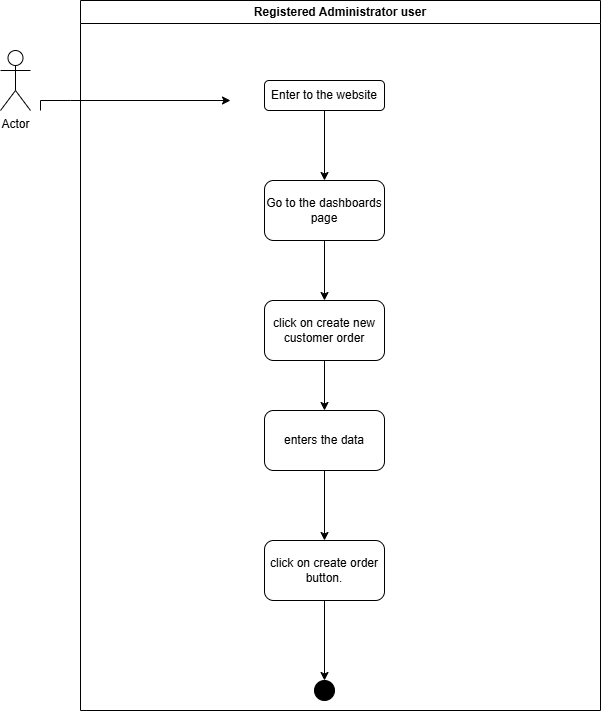


Figure 14 Activity Diagram for creating customer order.

## 4.3. Database Design

A relational model organizes data within a database into tables. Databases utilizing this model are known as relational databases. While these databases often use SQL as their language, SQL databases are not always considered relational databases. Despite being a decades-old model, the relational model remains beneficial for data organization. In this model, data is interconnected through the structure of the tables, ensuring relevant data is linked together .In the relational database model, tables are used to store information about various entities .Also, there are three primary types of relationships: one-to-one, one-to-many, and many-to many. Additionally, Cardinality refers to the minimum and maximum occurrences of an entity, while Connectivity refers to the type of relationship between tables .An attribute refers to what a column is called in a relational database . A UNIQUE key is a constraint that ensures one or more columns contain no duplicate values across different rows. A table can have multiple UNIQUE keys . An optional attribute, also known as nullable, can contain a NULL value . A primary key is a column or a set of columns in a table that uniquely identifies each row. Relational databases enforce the uniqueness of primary keys by ensuring that each value appears only once in the table . A foreign key is a column or group of columns in a table that matches the values of the primary key in another table. To insert a row with a specific foreign key value, there must be a corresponding row in the related table with the same primary key value .In the Relational Database Model, constraints are restrictions imposed on data or data processes. These restrictions ensure that only certain types of data can be entered into the database or that only specific types of operations can be carried out on the data. Constraints play a crucial role in maintaining data accuracy within a database management system (Relational Database Model) .

In the following page the entity relationship diagram of our project will be presented (figure 15), and since it is large, we chose to add it as a landscape to make it clearer. It consists of 10 entities, these entities have different types of relationships, consisting of many to many (between registered Users and the products, orders), and one-to-many (between registered User and suppliers).

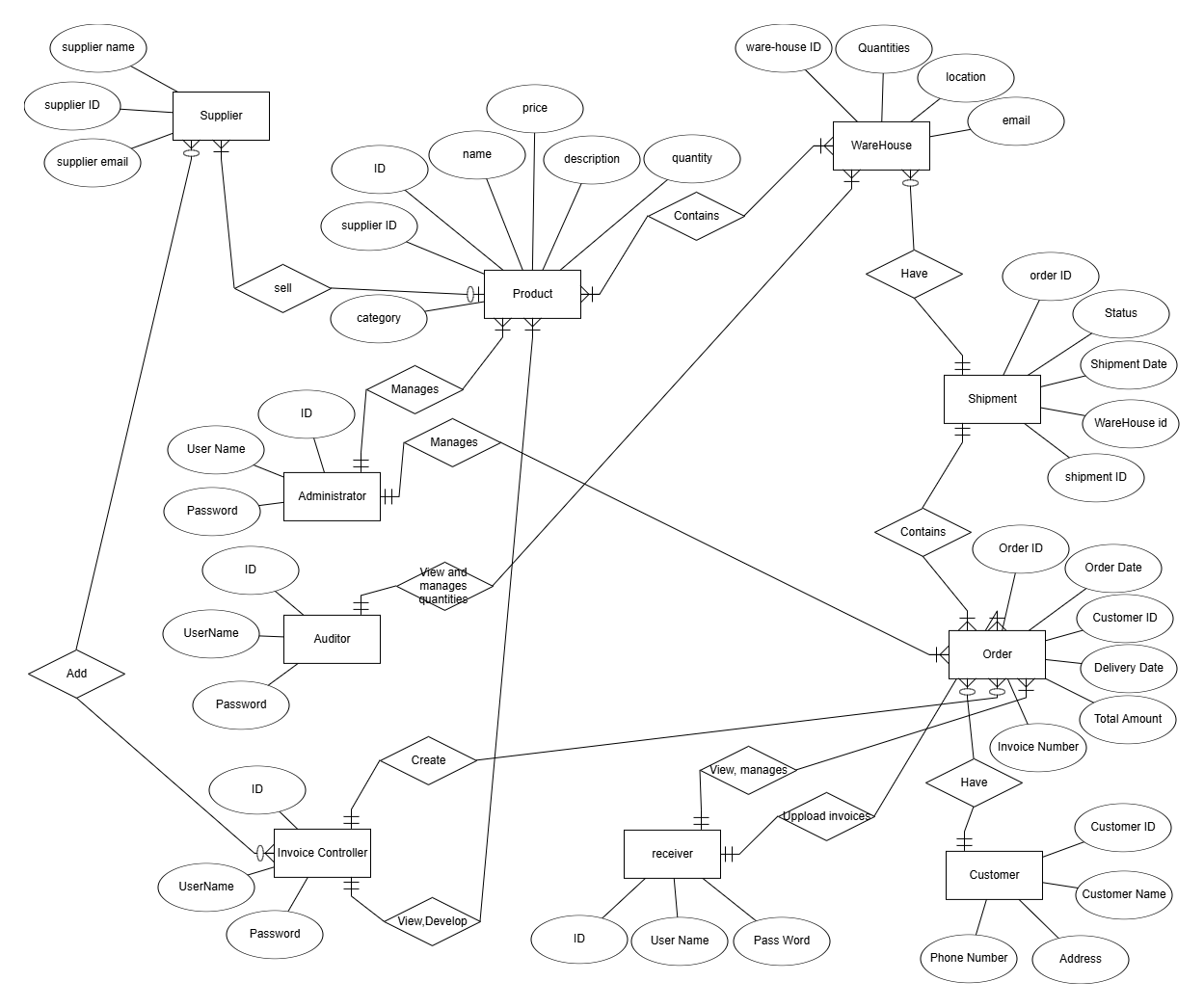


Figure 15 Entity Relationship Diagram

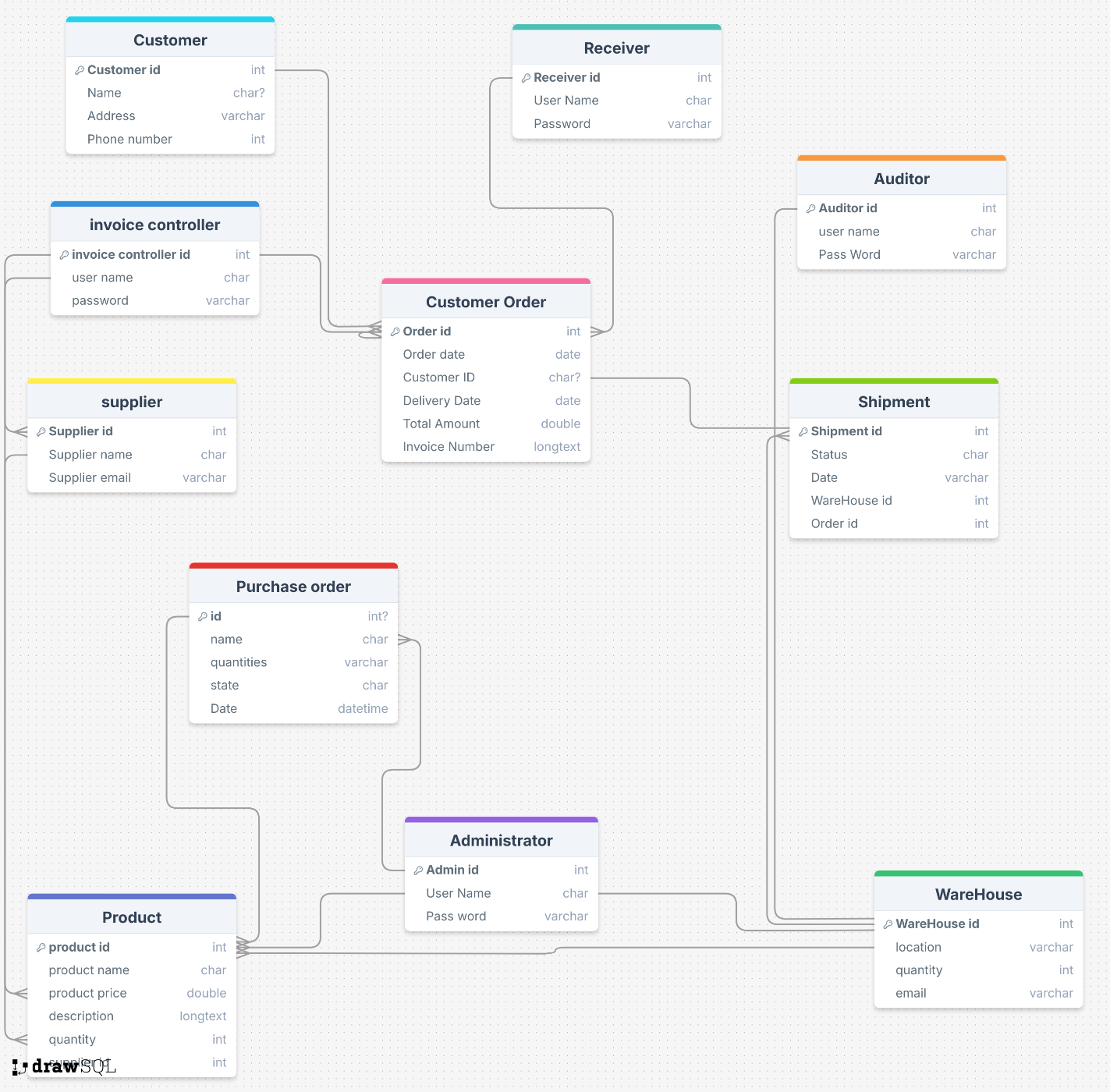


Figure 17 Relational Schema

This figure (figure 17) represents the relational schema. It shows specifically how the entities in the ER diagram are connected. And due to the many-to-many relationships, the primary keys, the foreign keys, the optional attributes, and the unique ones, new entities not shown in the ER diagram appeared in the relational schema.

## 4.3. Conclusion

In the previous chapter, the tools we used to help us develop our project were shown and explained, such as the use case diagram, activity diagrams, scenarios, entity relational diagram, and the relational schema. The use case diagram served as blueprint of the system, illustrating the interactions between actors that reveal the system’s functionality and requirements. Furthermore, the sequence of events and procedures for executing a feature or activity are depicted in scenario tables and activity diagrams. Relational Schema and Entity-Relationship diagrams are created to display the project’s structural framework, guarantee data integrity, and enhance storage efficiency.

**Chapter V: Application Implementation**

## 5.1. Introduction

In this chapter we will present all the technologies used to create our website as desired to make it extraordinary. We’ll let you have an overview of our website through several screenshots and some explanation. The main aim is to show you our special applications we included in this website.

## 5.2. Used Technologies

We were in need for several technologies and frameworks in order to make this website more user-friendly and interactive. These tools played the main role of serving the requirements with no issues to the users. We made sure to serve the users a flawless experience with the website in addition to making it eye-pleasing.

### 5.2.1. Used Backend Technology

We used Laravel framework version 8 with the JS language for the backend, and SQL Server for the database. This technology for the database offered a stable and reliable platform for managing, storing, and retrieving structured data.

In order to streamline our team's development process and manage version control efficiently, we made use of GitHub for the duration of our project. GitHub served as a hub for our team, enabling us to work together, handle modifications, and keep a consistent codebase efficiently.

### 5.2.2. Used Frontend Technology

Frontend development is a crucial aspect of web applications, and HTML, JavaScript, and Tailwind CSS are key technologies in this domain. HTML forms the structure of webpages, allowing for the creation and organization of web elements. JavaScript, a dynamic scripting language, adds interactivity to these webpages, enabling real-time updates and user event handling. Tailwind CSS, a utility-first CSS framework, enhances the visual appeal of the site, providing the tools for developers to build custom, scalable, and maintainable user interfaces. Together, these three technologies streamline the creation of effective, engaging, and responsive web applications, each playing a unique role in improving the user experience and the overall project effectiveness.

## 5.3. Application Screenshots

This section contains the main dashboards of the website.

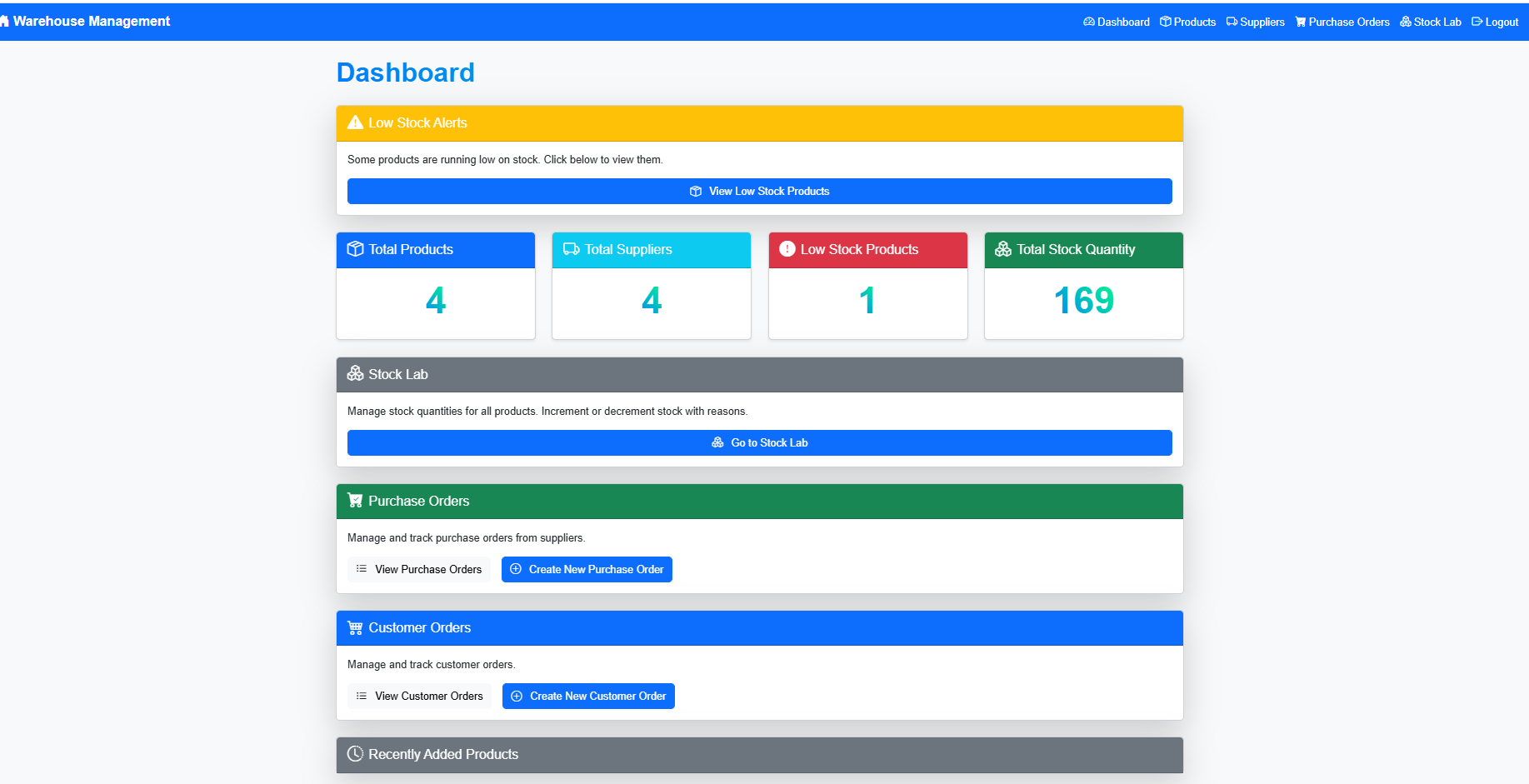


Figure 18 dashboard page

The above figure (figure 18) displays the dashboard page of our website where it displays all the total number of products in the stock, total number of suppliers, total number of low stock products, and the total stock quantities, also it displays the stock lab in order to go to the stock to edit quantities and more, also purchase orders in order to go to the purchase order page to create or edit any purchase order, it displays the customer orders and a button to go to the customer orders, finally it displays the recently added products.

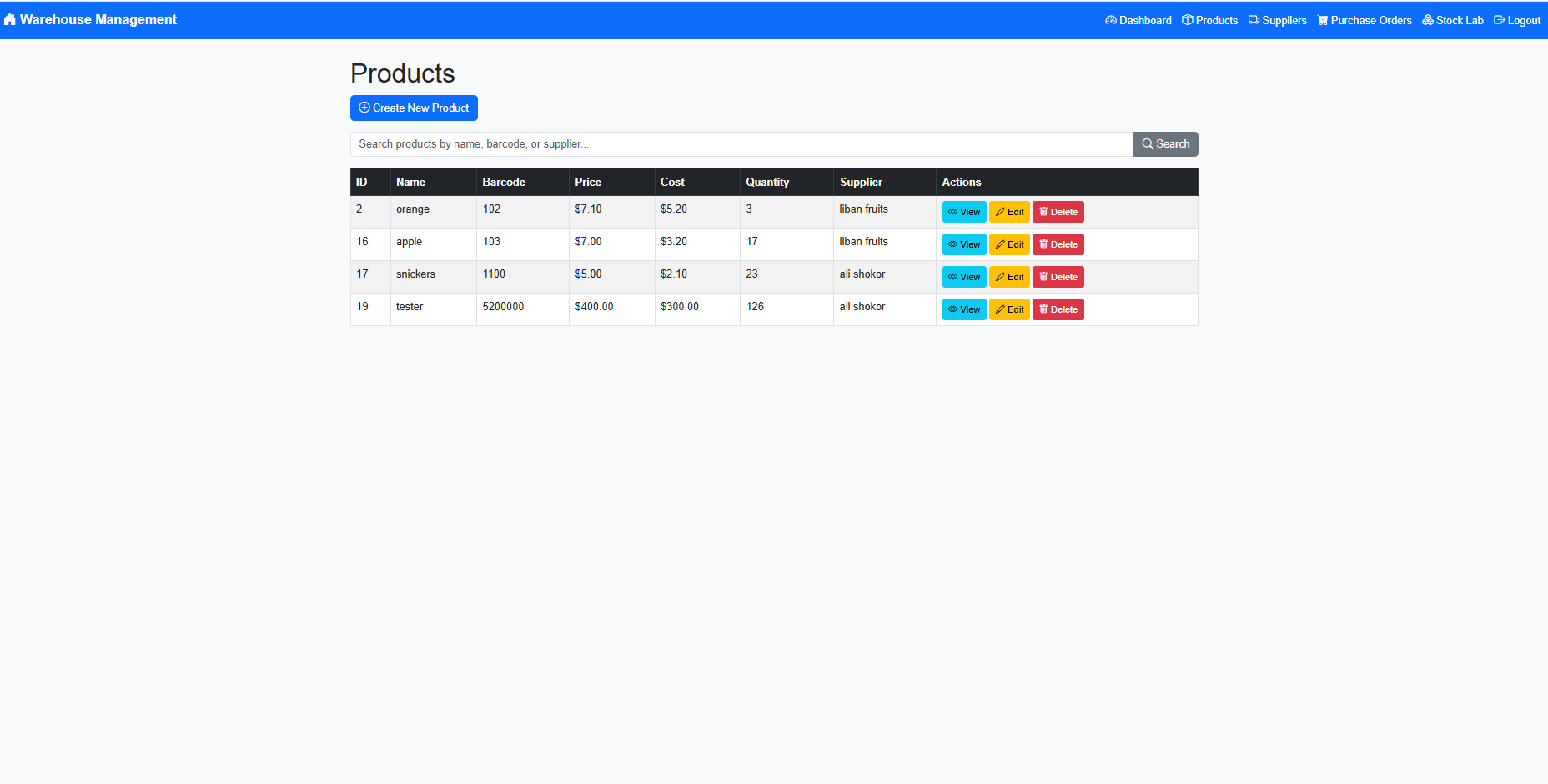


Figure19 table of products page

Now moving on the figure 19 which shows the products page where all products in the stock is shown there, and also in it we can create new product in the stock, and search for any product.

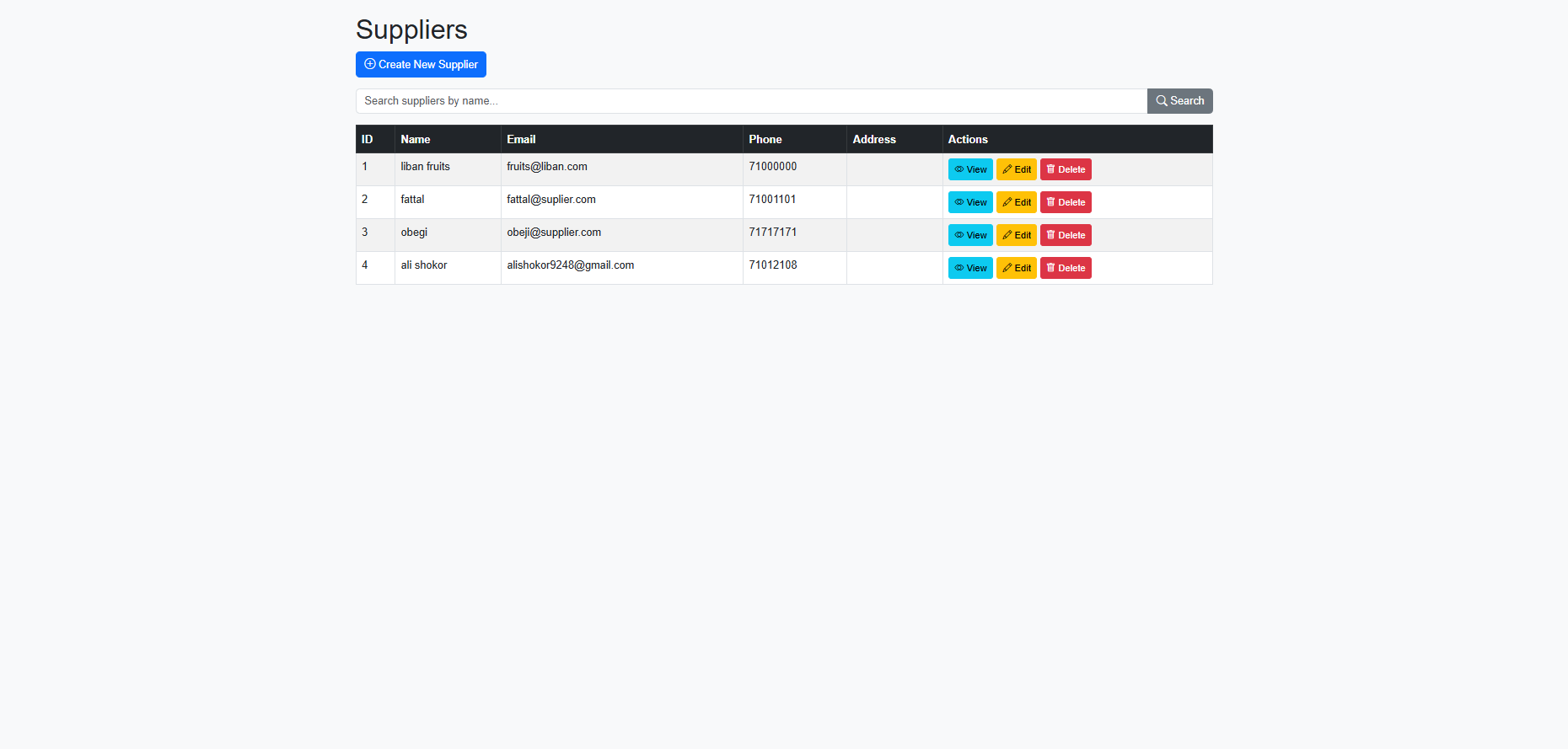


Figure 20 table of suppliers

The page in figure 20 contains the suppliers page where it contains a table containing all suppliers, and the control buttons on any supplier.

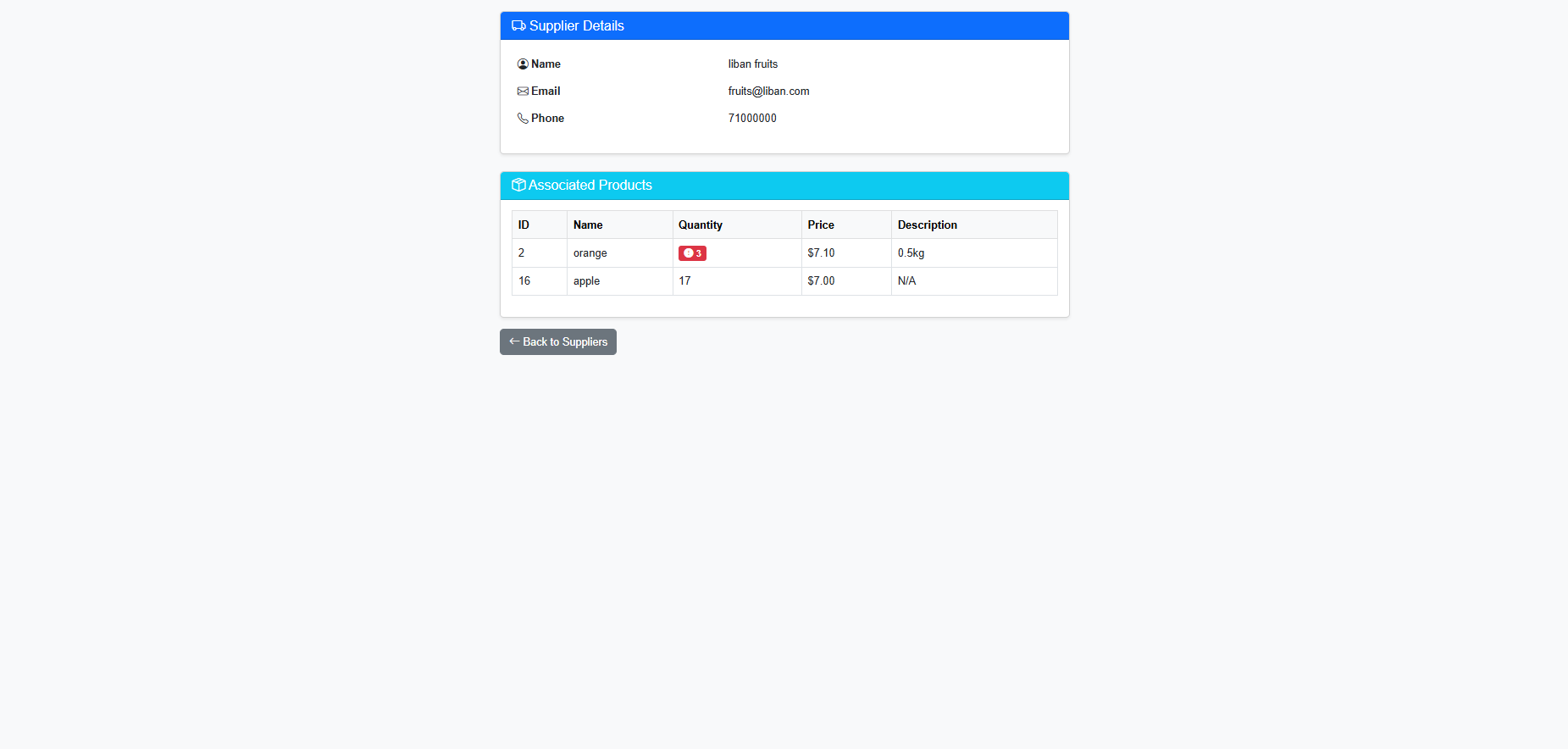


Figure 21 supplier page

Figure 21 having the whole details for a supplier (name, email, phone….) that can be edited also it contains all the items that a specific supplier can ship which is a feature that we only have

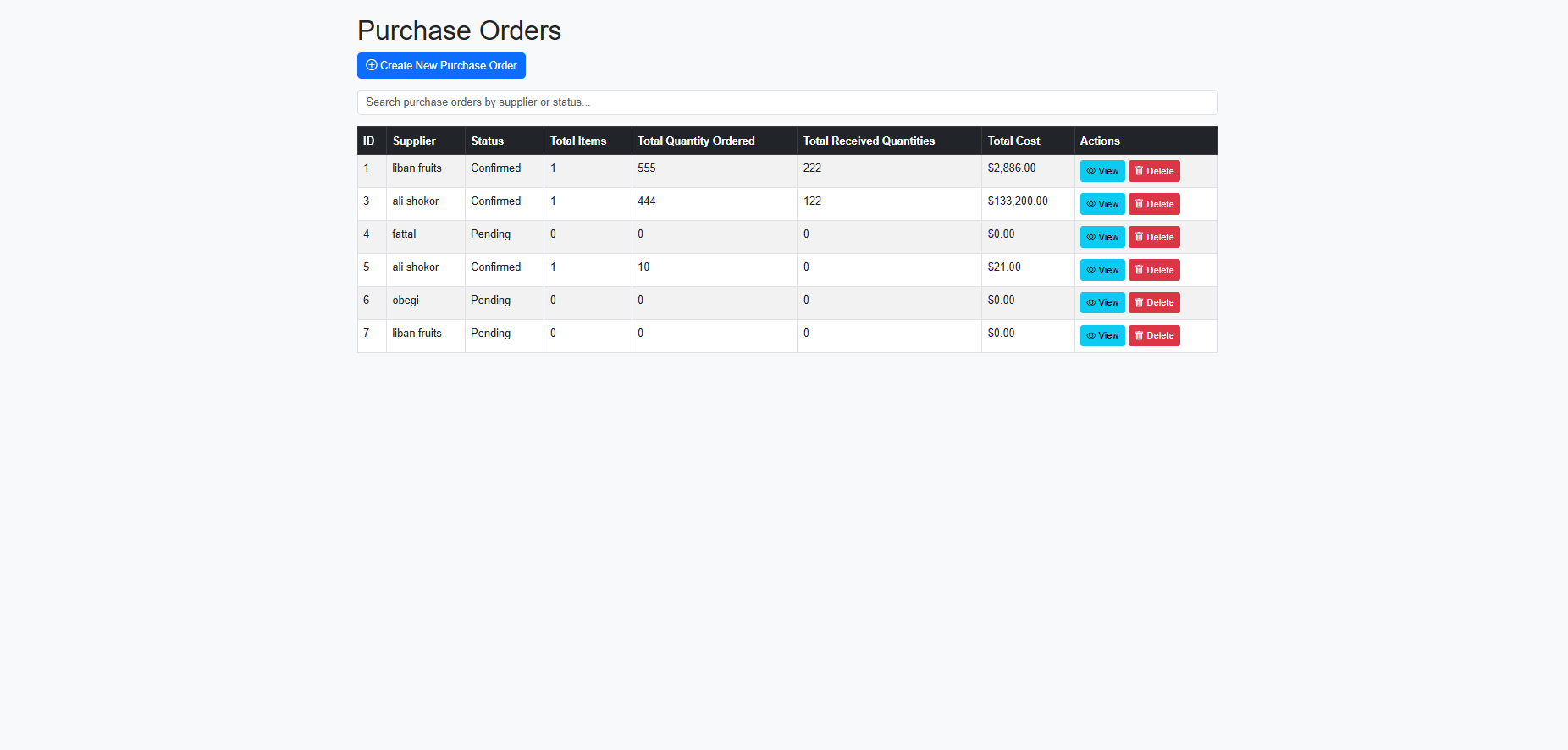


Figure 22 table of purchased orders

In the figure (figure 22), a table of the purchased orders, as well as a button in order to create a new purchase order.

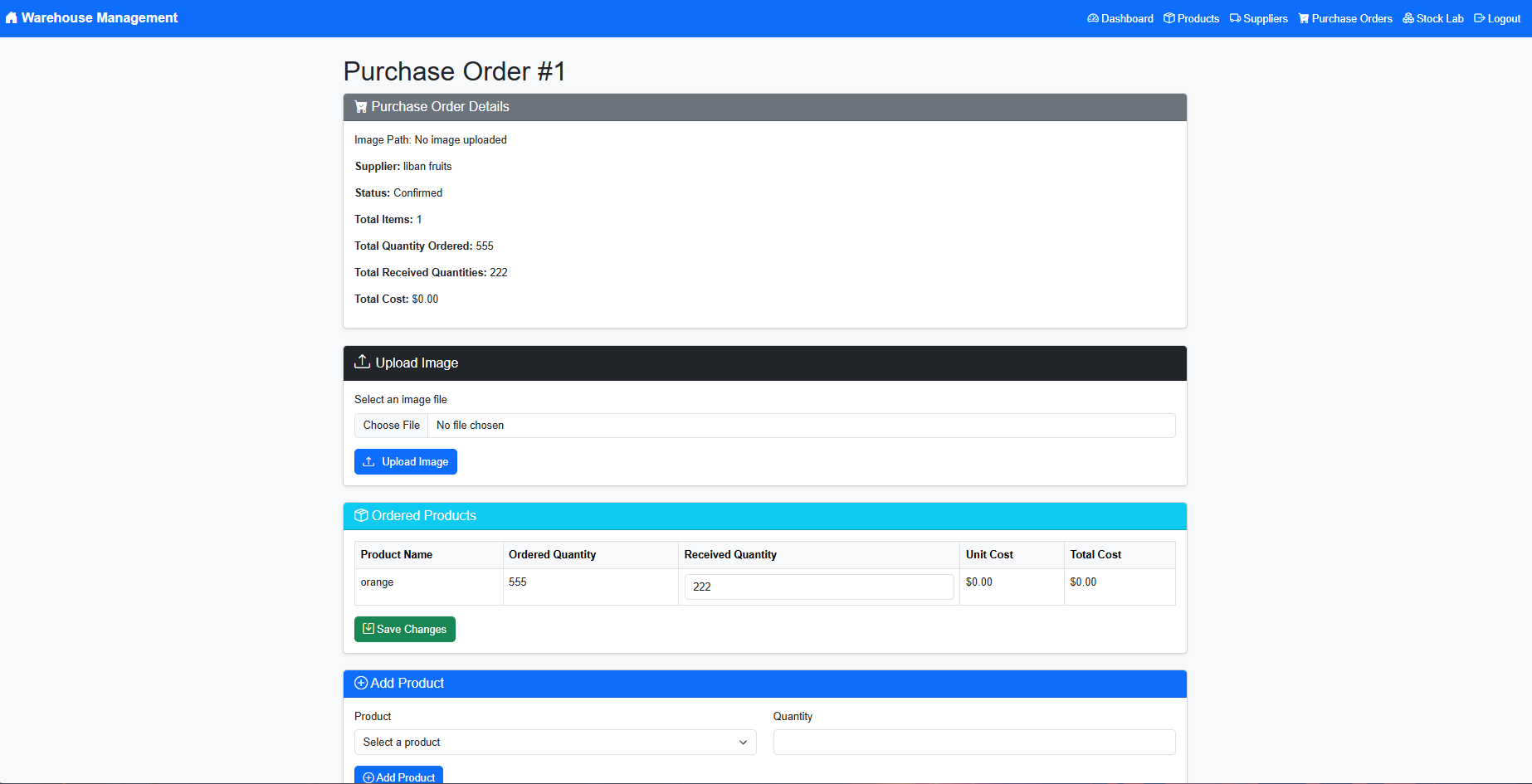


Figure 23 purchase order details page

In figure23 shows the whole purchase order data, (invoice image, ordered products, add product to the order…)

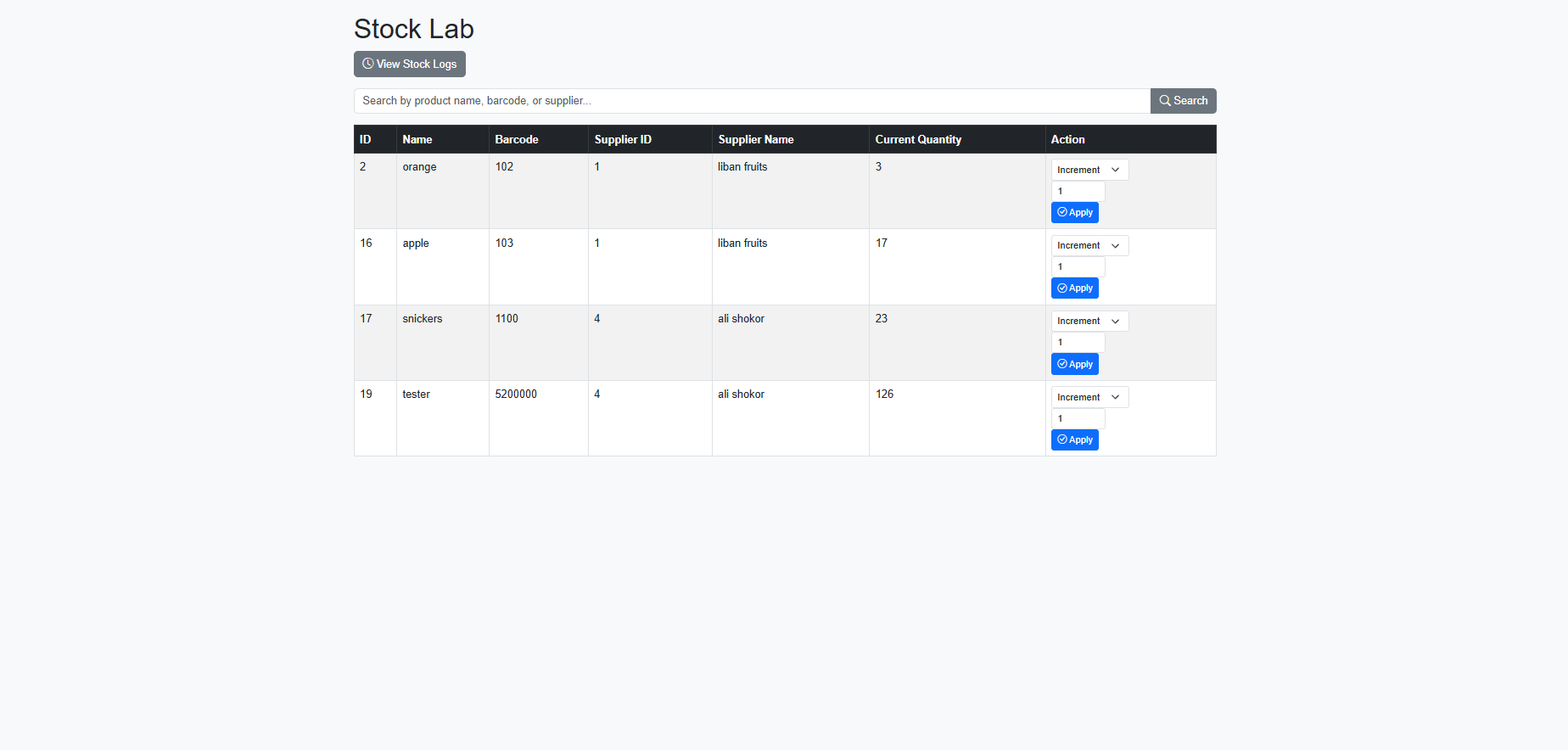


Figure 24 stock lab Page

figure 24 show a table containing every product details (name, barcode, supplier id, supplier name, quantity, and actions on every one)

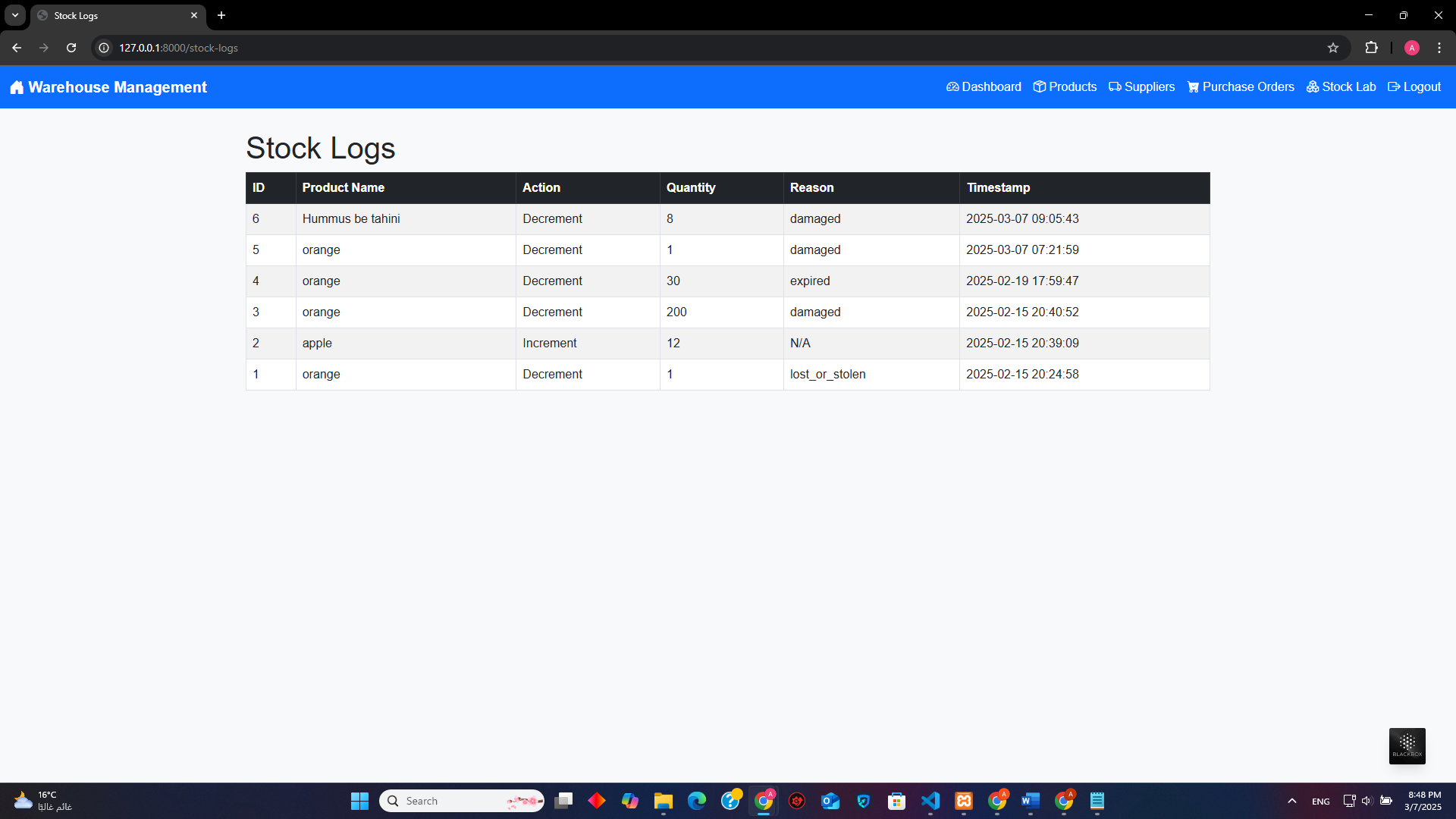


Figure25 stock logs

This figure shows the stock logs where what is decremented or incremented is saved to investigate about each

5.4. **Code Screen shots:**

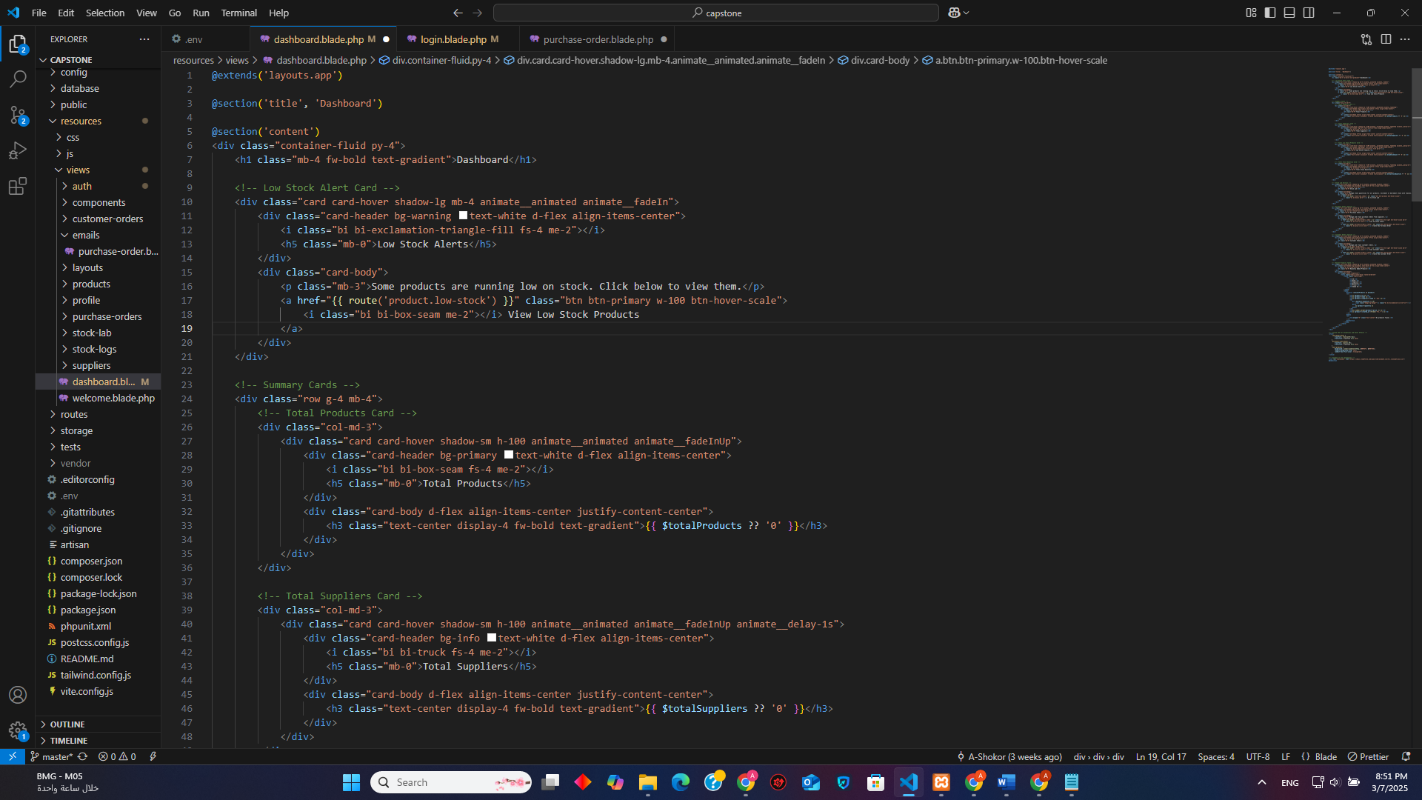


Figure 26 dashboard blade 1/2

The code creates a dashboard interface with two main sections: a Low Stock Alert Card and a set of Summary Cards. The Low Stock Alert Card uses a yellow header, a warning icon, and a button to draw attention to low stock issues, linking to a detailed view. The Summary Cards display key metrics (total products and suppliers) in a clean, responsive layout, with consistent styling, icons, and staggered animations for a dynamic feel. Both sections use dynamic placeholders ({{ $totalProducts }}, {{ $totalSuppliers }}) to show real-time data, defaulting to '0' if unavailable. The design emphasizes usability with clear visual hierarchy, hover effects, and accessibility features like semantic HTML and icon labels, making it both functional and visually appealing.

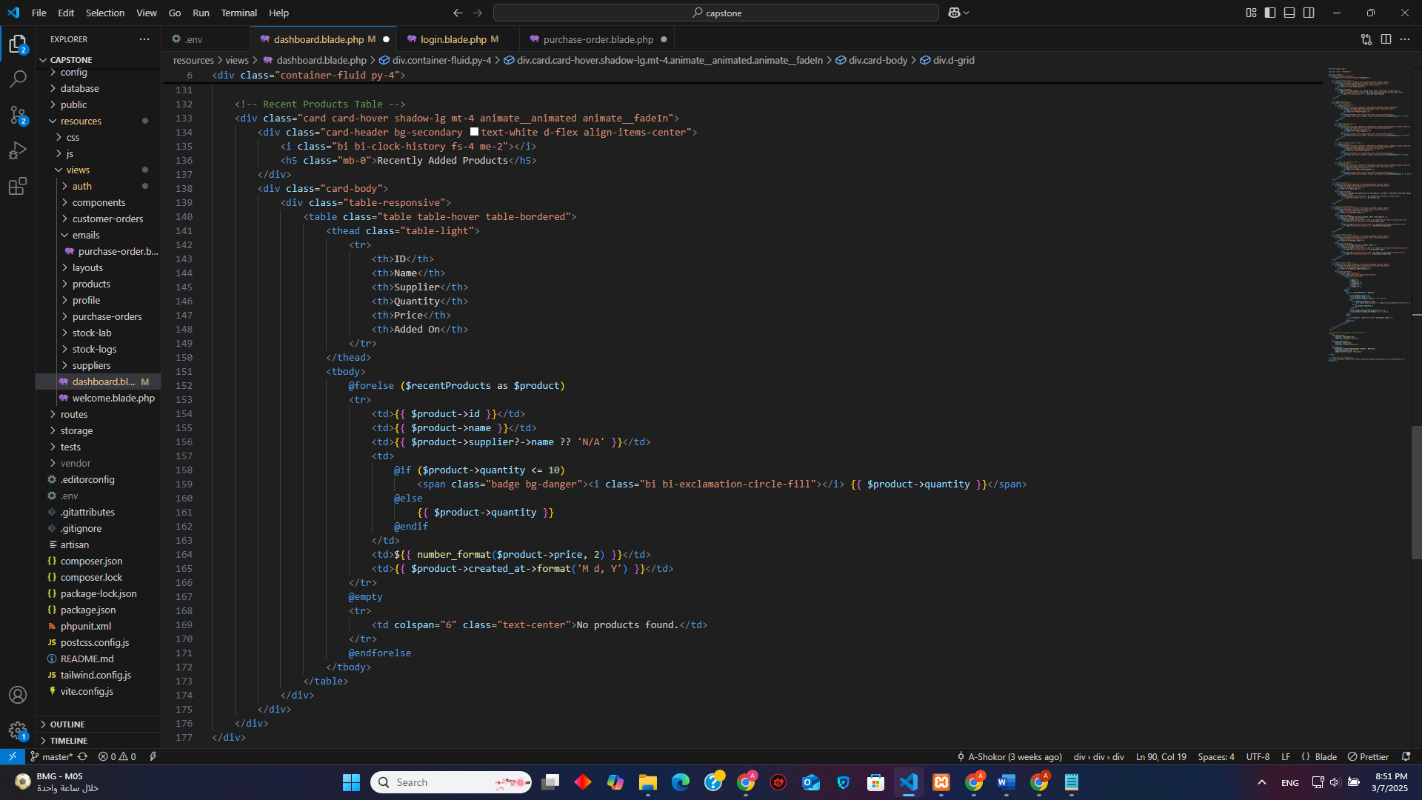


Figure27 dashboard 2/2

This code creates a Recent Products Table card that displays a list of recently added products in a responsive and interactive table. The card has a secondary-colored header with a clock icon and a title, emphasizing the recency of the data. Inside, a responsive table (table-responsive) lists product details like ID, name, supplier, quantity, price, and addition date. The table uses conditional formatting: if a product's quantity is low (≤ 10), it highlights the quantity with a red badge and a warning icon. If no products are found, it displays a "No products found" message. The card also includes a fade-in animation (animate\_\_fadeIn) and hover effects (card-hover, table-hover) for better user interaction. Overall, it’s a clean, dynamic, and user-friendly way to present recent product data.

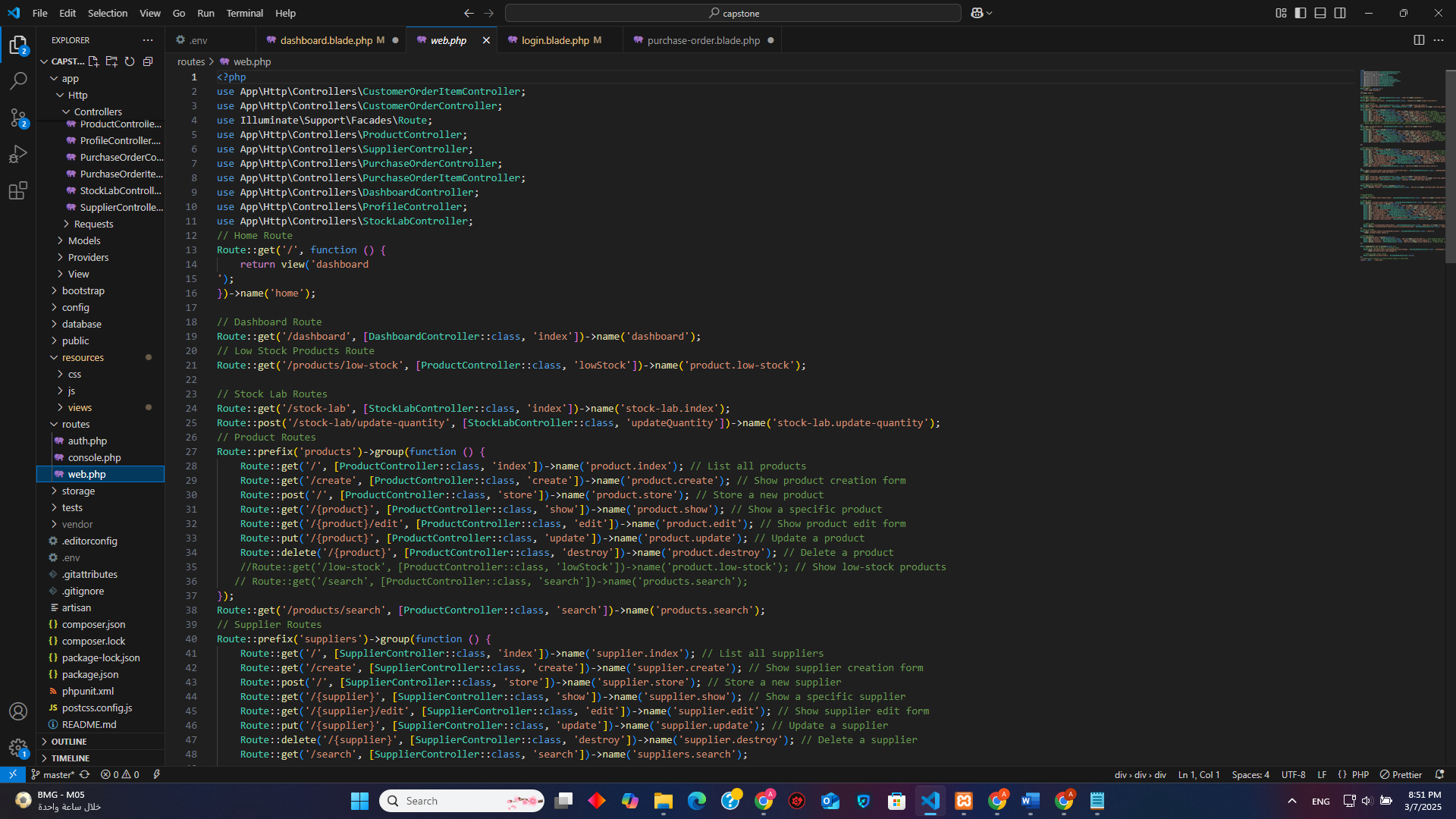


Figure28 routes1/2

This code defines a set of routes , organizing its navigation and functionality into clear, modular sections. The root URL (/) and /dashboard route serve as the application's homepage and main interface, while the /products/low-stock route specifically handles displaying low-stock products, likely used in the Low Stock Alert Card. The Stock Lab Routes (/stock-lab and /stock-lab/update-quantity) manage stock quantity viewing and updates. Product-related routes, grouped under the products prefix, handle CRUD operations (create, read, update, delete) for products, along with search functionality and low-stock product listings. Similarly, supplier routes manage CRUD operations and search for suppliers. Overall, the code provides a structured and efficient way to handle backend logic, ensuring seamless management of products, suppliers, and stock-related tasks.

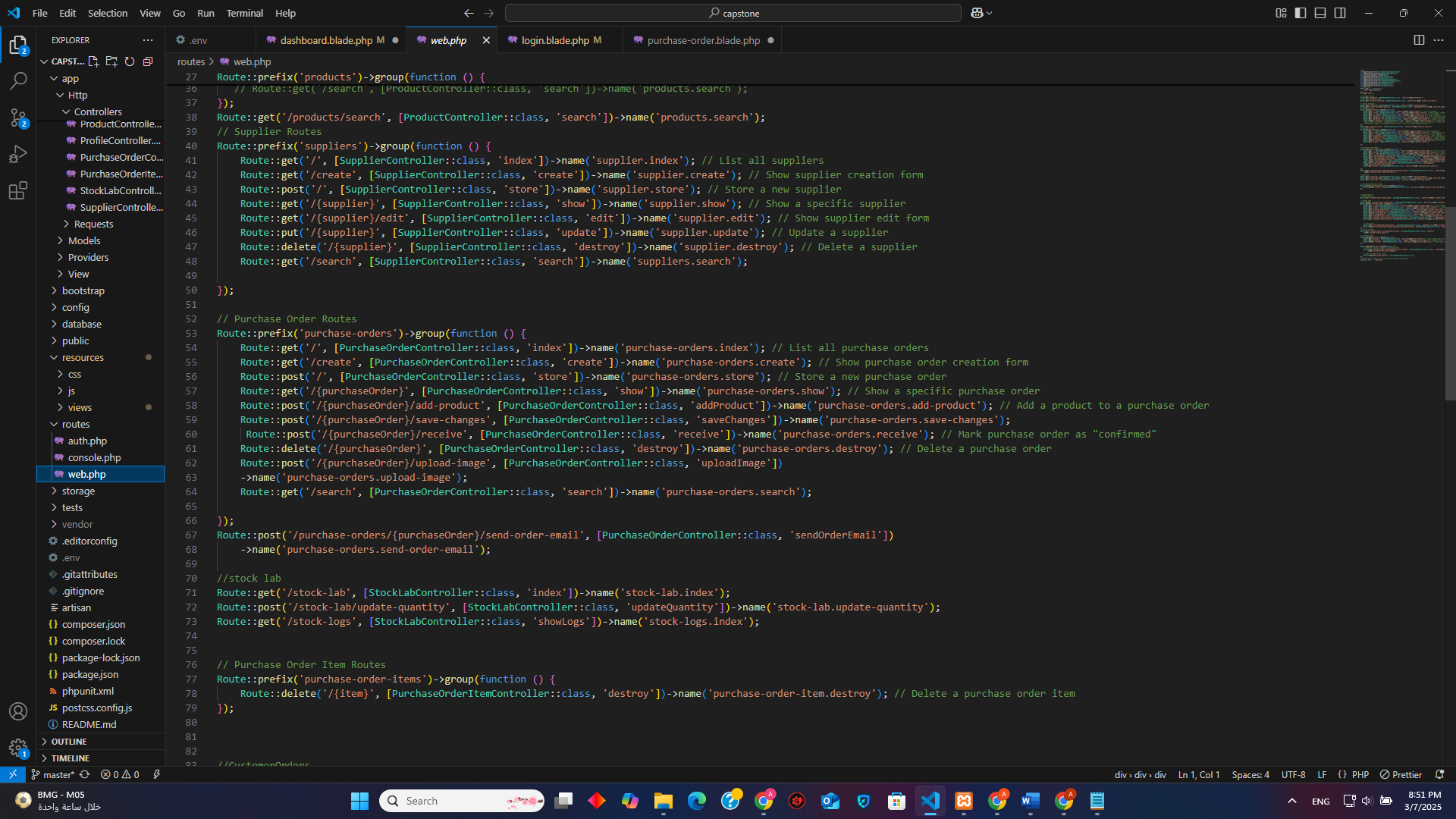


Figure29 routes 2/2

This code defines a comprehensive set of routes, organizing backend functionality into logical groups. The Purchase Order Routes handle CRUD operations for purchase orders, including adding products, saving changes, receiving orders, uploading images, and sending order emails. The Stock Lab Routes manage stock quantity updates and logs, ensuring accurate inventory tracking. Customer Order Routes facilitate the creation, editing, and management of customer orders, including adding/removing products, updating order statuses, and canceling orders. Additionally, Profile Routes allow authenticated users to edit, update, or delete their profiles. The code also includes authentication routes (via auth.php) and ensures secure access to certain routes using middleware. Overall, the routes provide a structured and efficient way to manage purchase orders, customer orders, stock, and user profiles, making the application robust and user-friendly.

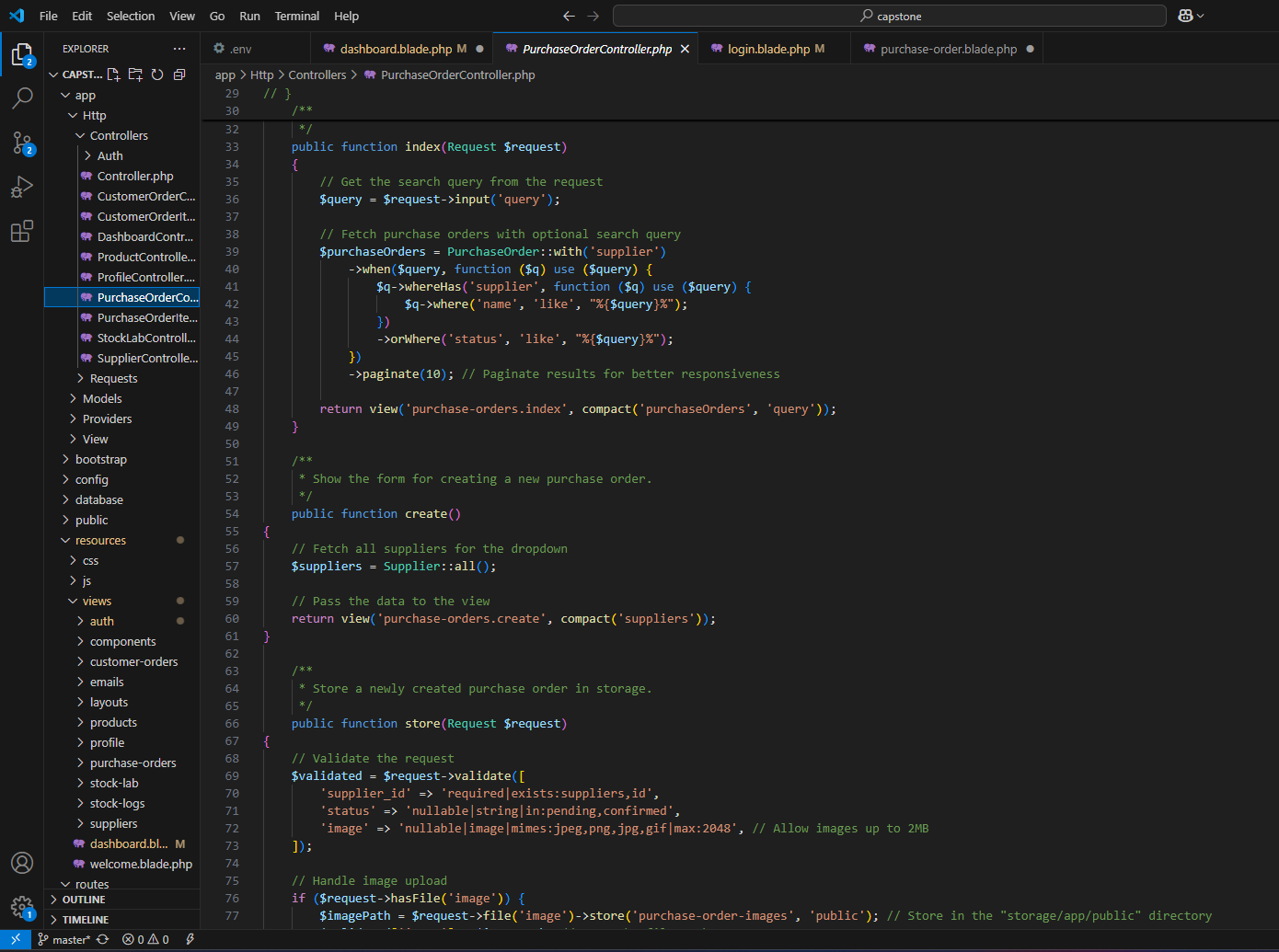


Figure 30 controller (po)

This code defines methods for managing purchase orders in a. The index method fetches and displays a paginated list of purchase orders, optionally filtering results based on a search query that matches either the supplier's name or the order status. The create method retrieves all suppliers to populate a dropdown in the purchase order creation form. The store method handles the creation of a new purchase order, validating the input (including an optional image upload) and saving the data to the database. If an image is uploaded, it is stored in the public storage directory, and its path is saved in the database. These methods ensure efficient and user-friendly management of purchase orders, with features like search, pagination, and image handling enhancing functionality and usability.

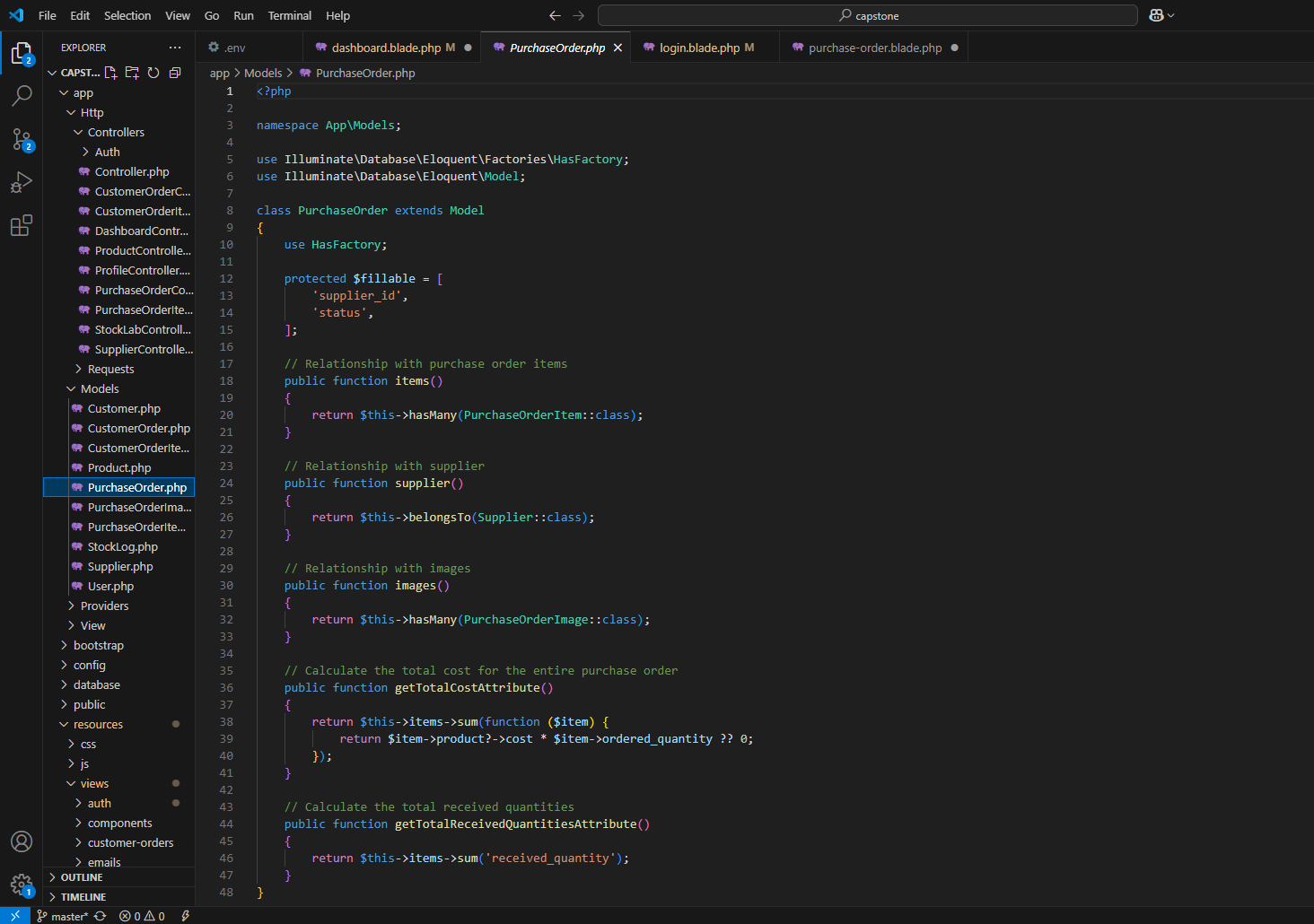


Figure 31 model (PO)

The code defines the PurchaseOrder model, representing purchase orders in the database. The model includes fillable fields (supplier\_id and status) to allow mass assignment. It establishes relationships with other models: a one-to-many relationship with PurchaseOrderItem (via the items method), a many-to-one relationship with Supplier (via the supplier method), and a one-to-many relationship with PurchaseOrderImage (via the images method). Additionally, the model includes two custom attributes: total\_cost, which calculates the total cost of the purchase order by summing the cost of all items, and total\_received\_quantities, which calculates the total received quantities of all items. These features make the model a central component for managing and querying purchase order data efficiently.

## 5.6. Conclusion

To conclude, in this chapter we mentioned the used technologies in our project. Then, the user interface of our website and the shining features were presented. Moreover, some of the snippets of code were included.

**Chapter VI : Conclusion**

In conclusion, the development and implementation of the warehouse system application have successfully addressed the core challenges of modern warehouse management, including inventory tracking inefficiencies, order processing delays, and lack of real-time visibility. By integrating features such as automated inventory management, uploading images, and seamless third-party system integration, the application streamlines operations, reduces human error, and enhances decision-making through data-driven insights.

The system’s user-centric design ensures accessibility for warehouse staff while providing administrators with powerful tools to monitor performance and optimize workflows. Testing phases and stakeholder feedback validated its reliability, scalability, and adaptability to diverse warehouse environments.

Though challenges such as data synchronization and user adoption were encountered during development, iterative improvements ensured the final product aligns with industry standards and user expectations.

Ultimately, this project underscores the transformative potential of technology in revolutionizing supply chain efficiency. By bridging gaps between manual processes and digital innovation, the warehouse system application not only meets current operational demands but also positions businesses for sustainable growth in an increasingly competitive market.